NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN INVESTIGATION INTO THE COUPLING
OF INTERACTIVE AND BATCH NETWORK SERVICES
IN COINS

by

Joanne Bong Soon Kim

June 1983

Thesis Advisor: N.F. Schneidewind

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Investigation Into The Coupling of Interactive and Batch Network Services in COINS		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Joanne Bong Soon Kim		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1983
		13. NUMBER OF PAGES 194
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Networks
Information-sharing
Batch and Interactive Coupling
Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Networks were conceived in the 1950's, born in the 1960's and grew up in the 1970's. Today they constitute a technology with applications in a myriad of disciplines. Information sharing has been one of the areas greatly aided by computer networks. The Community On-Line Intelligence System (COINS) is an information sharing network in the U.S. intelligence community. COINS offers batch and interactive services which

are separate and independent of each other. The information acquisition process has elements of interactive and batch. The design of an information sharing network should provide the foundation to accommodate this two-phased activity. This thesis introduces the concept of collaboration between these autonomous network services, proposes a re-allocation of network capacity in COINS and examines how this new scheme can improve performance and efficiency from a user and managerial perspective.

Approved for public release; distribution unlimited.

An Investigation Into The Coupling of Interactive and Batch Network Services in COINS

by

Joanne Bong Soon Kim

B.A., University of Hawaii, 1966

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(Command, Control and Communications)

from the

NAVAL POSTGRADUATE SCHOOL
June 1983

ABSTRACT

Networks were conceived in the 1950's, born in the 1960's and grew up in the 1970's. Today they constitute a technology with applications in a myriad of disciplines. Information sharing has been one of the areas greatly aided by computer networks. The Community On-Line Intelligence System (COINS) is an information sharing network in the U.S. intelligence community. COINS offers batch and interactive services which are separate and independant of each other. The information acquisition process has elements of interactive and batch. The design of an information sharing network should provide the foundation to accommodate this two-phased activity. This thesis introduces the concept of collaboration between these autonomous network services, proposes a re-allocation of network capacity in COINS and examines how this new scheme can improve performance and efficiency from a user and managerial perspective.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

In recent years, we have witnessed changes in information searching and sharing practices. With the dramatic decrease in digital technology and the concomitant advancements in computing and communications, we have seen the birth of the new technology of networking. Like any tool, which is viewed as a solution to a problem, so too networking and in its many forms has been brought to bear on a variety of problems [Ref. 1], [Ref. 2]. Information sharing has been one of the areas greatly aided by this new technology. It is now possible to have real-time interactive access to massive amounts of information around the globe at the touch of one's finger-tips. There are numerous information sharing networks in private industry, the Department of Defense (DoD) and other government agencies. One of these is the Community On Line Intelligence System (COINS) which interconnects on-line information storage and retrieval systems located at several locations within the U.S. Intelligence Community. COINS provides world-wide access to these information resources.

There was a point when the general belief was that all network access of the future would be interactive with a demise of batch processing. However, this view has been ameliorated after a close inspection of the user needs. Batch processing is still very useful and desirable. In many cases, a batch facility can enhance the analytic use of interactive services. COINS has both batch and interactive network facilities, but they are separate and independent. In each domain, we can envisage users attempting to use each facility to do both interactive and batch work. What is called for is some type of coupling of interactive and

batch network capabilities which matches users' needs. The purpose of this thesis is to introduce the concept of collaboration between otherwise autonomous operations and to study a scheme reflecting this synergetic notion. We will examine what effects this has on performance and efficiency from a user and manager perspective. We have designed and implemented a computer simulation of the flow of user requests to the interactive facility in COINS to help study the merits of the two approaches.

This thesis is organized as follows:

1. description of the COINS network, its architecture and its current implementation techniques for interactive information sharing;
2. discussion of the evaluation criteria for network performance;
3. presentation of an alternate proposal with discussion of how this new scheme is likely to improve interactive information sharing;
4. description of the interactive network services simulation model;
5. discussion of preliminary results using the model;
6. discussion of the simulation model's applicability in evaluating an alternative capacity allocation strategy as COINS grows; and
7. conclusions and recommendations.

II. THE COINS NETWORK

A. BACKGROUND

The Community On Line Intelligence System was establised on the recommendation of the President's Foreign Intelligence Board (PFIB) in June 1965 to improve information handling methods. The implementation plan called for a star-configured network to provide connectivity among the intelligence data processors. The concept was to permit an analyst sitting at his local terminal to access information either at his host processor or at a remote central processing unit (cpu). The participants were the Central Intelligence Agency (CIA), the Defense Intelligence Agency (DIA), the National Security Agency (NSA), the National Photographic Interpretation Center (NPIC), the State Department, and the National Indications Center (NIC). The store-and-forward message switching node was physically located at DIA. Figure 2.1 shows the original COINS configuration. Implicit in this concept was the requirement for an intelligence organization to have a cpu connected to the COINS-switch to access information in COINS. The State Department and NIC did not have cpus in COINS. Hence access for these two organizations and any others that did not have cpus was by procuring a terminal from one of the host processors in the network.

Each of the nodes offered the same batch query and retrieval services to the network as they did to their local users. Users would submit their network queries at their local terminals and some time later would receive their responses. Depending on the data manipulation tools of the host, the responses would range from simple data record

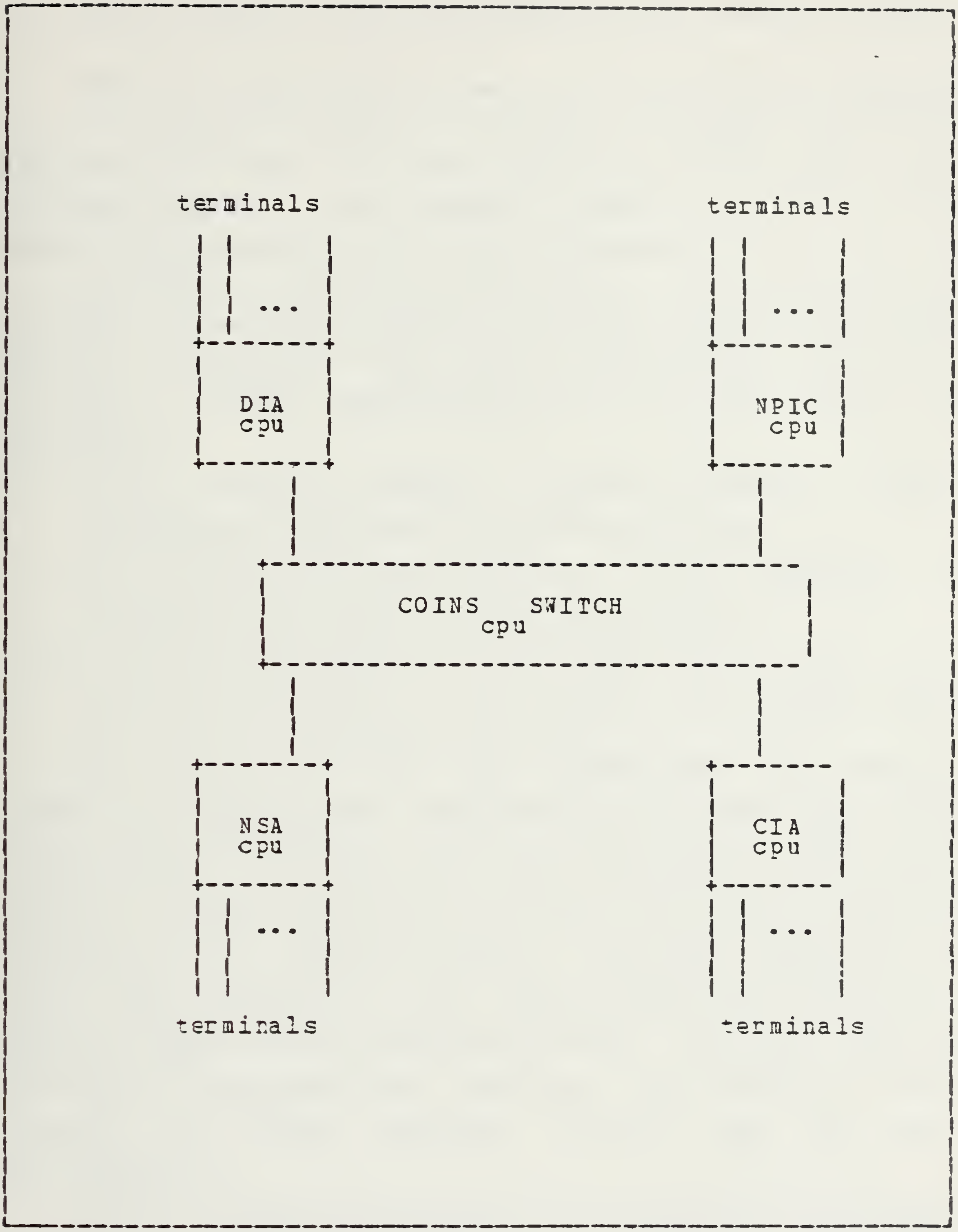


Figure 2.1 Original COINS configuration.

listings to some statistical summaries of numeric information.

Over the years COINS access and participation expanded with a netting of several military commands under the management of DIA. It was called the Intelligence Data Handling System, Communications (IDHSC). Its form was simply the extension of the star-configuration.

By 1975, significant developments in teleprocessing provided the necessary impetus for COINS to move to the next chapter in networking with the introduction of new network services. COINS assimilated into its architecture the following:

1. the Advanced Research Projects Agency's (ARPA) networking technology of packet-switching;
2. the Sigint On Line Information System (SOLIS), an interactive, full-text retrieval system;
3. a user-Terminal Access System (TAS); and
4. Front-End processors which connect the database cpu's to the new networking technology.

The concept of a TAS was necessitated by a growing number of intelligence organizations without cpus that wanted access to COINS. This requirement was further reinforced by the private sector idea of relieving the database hosts of terminal handling functions and putting all user interfaces on a separate facility. The TAS provides both batch query services to the batch hosts and interactive query services to the interactive host. With the adoption of this new technology and services, the network was named COINS-II. The IDHSC component of COINS kept the star-configuration.

In 1976, COINS-II undertook an internetting experiment with the ARPANET, installing a TAS in Hawaii. As a result the Pacific Command (PACOM) now has secure interactive access to the full-text retrieval system.

By 1980, COINS-II introduced a new kind of TAS. While the original TAS is a pure user which does not offer any databases to the network, this new TAS has on-line user support functions and network management information. This type of TAS will be denoted as a server-TAS.

B. CURRENT SYSTEM

Figure 2.2 is a picture of COINS-II today. Currently there are two TASs, two server-TASs, one interactive, full-text retrieval host, and five batch retrieval hosts. The user TASs are called TAS and AKU. The first TAS retained the name TAS. The server-TASs are called NSH (Network Service Host) and TRF (Transfer Research Facility). With the COINS/PMO developing a family of TASs, the NSH has evolved into a MASTER-TAS, similar to the concept of the MASTER-IMP in ARPA technology whereby software releases and remote debugging are done. NSH also supports a small cadre of operational users. TRF is the COINS's developmental facility where research ideas can be developed and tested in an operational environment. The user-support system resides on TRF. No intelligence analysts are supported by TRF. The database hosts have Front-Ends (FEs) to connect them to the communications network. Each retrieval system has its own language and each data file has its own coding schemes. As part of the ARPANET technology, there is a Network Control Computer (NCC) for the communications network monitoring and management.

C. FUTURE PLANS

With respect to network growth, the COINS/PMO anticipates two more interactive server-hosts and four more TASs by 1985 [Ref. 3]. In the area of network services, the COINS/PMO has a joint effort with the Department of the Army

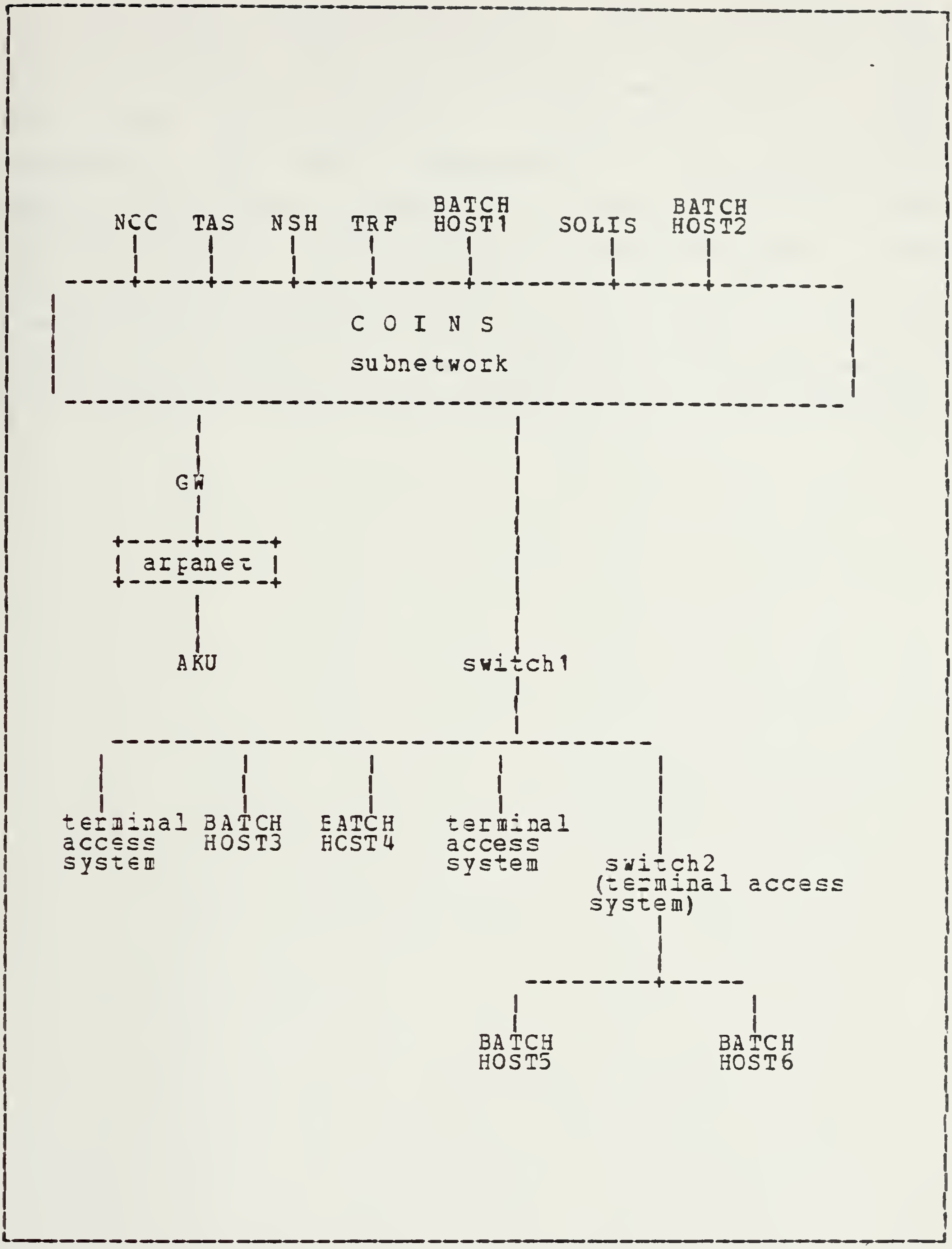


Figure 2.2 Current COINS-II configuration.

in data fusion analysis [Ref. 4]. For the multiple query language problem, COINS is continuing an effort called ADAPT which is a network language that users would employ. ADAPT would make the appropriate transforms to the target languages. The COINS organization is working with the Center for Computer Security at NSA on the multi-level security problems in networks and in inter-networks. COINS also has several ARPA-sponsored efforts in the area of human-factors engineering for the network user. Here the concern is with the work environment and the development of an intelligence analyst work-station of the future [Ref. 5].

III. NETWORK CONFIGURATION

Network configuration is a combination of communications, hardware and software. These components are discussed in this chapter.

A. THE SUBNET ENVIRONMENT

When the term "sub-network" is used below, it will mean the communications technology supporting host connectivity. The sub-network configuration of COINS-II is the packet-switching technology of ARPANET. The six IMPs (Interface Message Processors) are distributed over five sites in the Washington, D.C. area and are connected by 56 kilobit/second phone lines. The IMPs are a mix of Honeywell- 316s and C-30s. With the use of a special gateway (GW) and a pair of private line interfaces (PLIs), COINS-II is internetted with the ARPANET, resulting in connectivity to Hawaii. There are 15 hosts in COINS. There are seven intelligence database cpus (one interactive and six batch) and seven terminal access systems (four TASs as described in the previous chapter, three developed under IDHSC). The last host is the Network Control Computer.

B. HARDWARE ENVIRONMENT

At the lowest level, the subnet capacity is 56 KB/s. This is the physical maximum data transmission rate of the telephone lines. The boxes contributing to the effective throughput are the communications crypto equipment, the IMPs, the front-end processors and the hosts. Each of these components have related software to make them perform their network functions, and in the case of the cpus their user services functions.

During the installation testing phase of the packet-switching subnet in 1972, effective data throughput measurements were in the 28-32 KB/s range. The configuration for this testing included PDP-11 cpus connected to the IMPs. All the PDPs had an LH/DH interface unit that permitted connection to their IMPs according to the BBN 1822 specifications for interconnection of a host to IMP. Each of the cpus had a simple data generation program. The range of the throughput measurements was attributed primarily to the differing cpu capacities of the PDP 11/40s, PDP 11/45, and PDP 11/70. The next testing level had the same physical configuration but different software environments. The PDP 11/40's were running under ELF-I, the PDP 11/45 was running under RSX-11 and the PDP 11/70 was running under UNIX. For each of these systems, there was an Network Control Program (NCP) to handle the host-to-host protocol as specified by the ARPANET. There was also an application program that served as a data generator. The effective throughput measurements from this test were in the range of 15 to 20 KB/s. All of these throughputs excludes the host-to-host protocol overhead.

Since these measurements were taken, two server-TASs and one TAS were added to COINS. The TASs are configured to handle 16 to 64 terminals. These are Teletype Model-40 terminals with CRT, keyboard and printer, and operate in full-duplex at 2400 baud. The printer is slaved to the CRT. The one data-receive line is directed to the CRT.

C. SOFTWARE ENVIRONMENT

The software environment may be viewed as a trinity consisting of the operating system, the application software and the network software. At system generation time, these three components define the number of ports with which the

cpu will perform network business. For example, a certain amount of memory is allocated for the operating system, the application software and the networking software. In support of network services, a specific amount of system buffers is allocated. The number of system buffers in turn defines the number of simultaneous network connections a cpu can handle. Currently all TASs, including the server-TASs have an interactive network capacity of 24 ports. SOLIS, the interactive database resource in COINS, has a network capacity of 15 ports.

IV. USER SERVICES

SOLIS, NSH and TRF offer interactive access. The remaining six intelligence database hosts provide only batch access. Both batch and interactive are discussed in this chapter.

A. BATCH SERVICES

Use of batch query systems involves a user inputting the query at the local host and sending it to the remote system. He then receives a job number or receipt for the query. Some time later, which can range from minutes to days, the query response is delivered to the terminal. Responses are presented to TAS users only if they are logged on to a TAS and specifically request to see the response.

B. INTERACTIVE SERVICES

The three funtional interactive services available in COINS-II today are: intelligence database (SOLIS), user-support databases (USIS), and managerial and administrative databases (NUIS) as explained below.

1. Intelligence Databases

a. General

SOLIS is a partially formatted full-text search and retrieval system. It contains the last 13 months of messages and reports produced by the intelligence organization. Searching can be done on the formatted fields and on the full text in any combination with the normal boolean operators.

b. Exchange Discipline

The data exchange between SOLIS and the user is full screen. After successful logon, the user is presented with a form-screen where he can fill in the blanks with his search terms for the full text and the formatted fields. The user has three different form-screens to chose from. They are the AND-screen, the OR-screen, and the FREE-screen. In the AND-screen, the search terms are ANDed together; while in the OR-screen, the search terms are ORed together. In the FREE-screen, the user may compose his own boolean logic of the search terms. Within the AND- and OR-screen, use of parenthesis is permitted to achieve combinations of ANDs and ORs. After he has completed filling-in-the-blanks, the query screen is then forwarded to SOLIS. The response from SOLIS is again a full screen with the query the user sent plus the number of messages that satisfied the search. There is a blank area for the user to fill in, telling SOLIS what should be done next. At this stage, the user can:

(1) refine or modify the query and send that again;
(2) request display of titles only;
(3) request display of the formatted fields of the messages;
(4) request display of full-text of the messages;
(5) request a new form-screen;
(6) terminate the session; or
(7) request forwarding of titles only or titles and formatted fields or complete messages.

2. User-Support Databases

a. General

The User Support Information System (USIS) offers a variety of help functions to the network user. It is an attempt at Computer Aided Instruction (CAI) but not in

the same manner as the University of Illinois's PLATO system. While PLATO interacts with the user posing questions and checking the user's answers, USIS is primarily a one-way communication to the user. In USIS, the user can ask for one of several tutorials on the various languages and data files in the network and learn how to employ TAS commands and functions. While in USIS, the user can send messages to and receive messages from User-Central concerning aid he can't find in USIS, problems encountered and complaints. There is a limited amount of browsing and no refinement of output capabilities.

b. Exchange Discipline

USIS initially sends the user a menu of the available USIS-commands and waits for a response. The exchange rule from the user to USIS is one character at a time. First-level commands are normally terminated with a NEWLINE. Lower-level commands are terminated with a special sequence of period (.) and NEWLINE on a line by itself. Having reached this point, exchange between USIS and the user is as follows:

(1) USIS sends a screen of data;

(2) user may respond with browse command of forward, backward or return to the previous level;

(3) request for hard-copy must be done before reaching this level. The user must indicate it on his first command line.

The tutorials are basically copies of the hard-copy language manuals and file guides. They contain sample queries and outputs, including badly constructed queries and their resultant error messages. It also contains a file of latest-happenings in the network of interest to the user. An on-line newsletter is not available yet.

3. Managerial and Administrative Databases

a. General

The Network Management Information System (NMIS) contains statistics on file usages, number of queries, number of aborts, size of responses and network problems. It provides some basic matrix and chart displays. There is limited browsing and no output refinement capabilities.

b. Exchange Discipline

The data exchange between NMIS and the user is character at a time, which is terminated by the NEWLINE key. After successful logon, NMIS presents a menu to the user and asks for a response. After menu selection, NMIS prompts the user for search and display (viewed interactively or forwarded) criteria. This kind of exchange continues until NMIS has enough information to proceed with the actual work.

V. METHODOLOGY

This thesis is that reconfiguring the COINS interactive capacity into one consisting of collaborative interactive and batch functions can be, under certain conditions, superior to the current form. We plan to describe the nature of customer activity with a database as a combination of interactive and batch. It would make sense for the network to accommodate this two-phased activity in a manner that offers the best performance from both user and managerial perspectives. The examination will be confined to the work profile of the intelligence analyst. This has been prompted by the abundance of empirical data in this area and the very little information available for USIS and NMIS activities. The procedure outlined here can also be applied when investigating user support and managerial activities. We will then present a particular reconfigured network capacity scheme whose performance will be compared with the current method.

The available empirical data regarding TAS-customers' usage of SOLIS was gathered from COINS. The data was analyzed and statistical tests were performed to determine the underlying distributions of arrival rates and service times. The work-profile of a 'typical' network SOLIS customer was derived from SOLIS logs. TAS logs provided data on the percentage of user requests which required SOLIS access. Appendix A contains the results of this analysis.

We implemented a computer simulation to aid in the comparative analysis. The distributions developed from the data were used to drive the stochastic models of the various system configurations. Sensitivity analysis will be performed with respect to arrival rates and transaction service times. These two parameters were chosen because of

the expectation by the COINS/PMO of increased customer population and an in-agency study done on the SOLIS printing requirements. This study's conclusion was that there will be continued growth in demands for hard-copy output [Ref. 6].

Measurements will be made of

1. average system time which includes service time and wait time;
2. expected customer loss; and
3. proportion of interactive work.

The first two measures are directly related to customer satisfaction while the last may be of more interest to the network and database managers. Users are interested in the amount of time to accomplish a job. This is the total system time which includes both the service and wait time. They also anticipate an available server when they arrive for service. If the facility is busy when the customer arrives, he is lost to the system. If the facility is busy too often when the customer arrives, it will discourage system use and cause severe customer dissatisfaction. Although managers are ultimately interested in customer satisfaction, they also focus their attention on utilization issues. We propose a proportion measure with respect to interactive work. When evaluating an interactive query resource, it is important to scrutinize and ascertain how much of this facility is being used for interactive searching. Putting it another way, we would like to know how much of this facility is being used for non-interactive processing, that is, batch work. Hence the proportion measure will give the percentage of session time which is used for the search and refinement process. The complement of this is the proportion of time used for batch work.

VI. USER PROFILE ANALYSIS

A. GENERAL

Our model of network structure involves cpu to cpu connectivity as opposed to terminal to cpu connectivity. This thesis is an examination of a network form which involves cpu to cpu connectivity for interactive information sharing. We feel this is an important difference because we are dealing with a terminal interface that is capable of intelligence, capacity and speed far superior to that of a dumb terminal. The old adage that a chain is as strong as its weakest link is the same as saying that two devices can communicate as fast as the slower of the two. The same holds true for the other two attributes of capacity and intelligence. With respect to resource content, the commercial database systems are bibliographic and abstract in nature. This thesis examines usage of a full-text retrieval system. Wigington has suggested that searching full-text of large documents may have a somewhat different pattern from the bibliographic environment [Ref. 7]. This simply means we cannot take full advantage of the work already done for the commercial database system with respect to detail work-profile analysis. We will refer to these reports merely to give credence to our perception of the dichotomy of service-time.

User activity must be described in terms that will make it meaningful to the problem statement, i.e. non-interactive use of an interactive resource. Customer service-time may be classified into one of two modes. The first is interactive, and is defined as a continuing dialog between man and machine. When a user enters a request, he must wait to see

the system response before proceeding to the next request. The other is called non-interactive and is defined as follows: when a user enters a request for service, he need not wait to see the system response before he can proceed to the next request for service.

There is a plethora of studies on customer activities with commercial and non-Defense interactive network information resources and only a very few relating to full-text retrieval databases. We will refer to these commercial and non-Defense reports to show that we are correct in our perception of the user work profile. These investigations thus far have concentrated primarily on the interactive nature of bibliographic searches [Ref. 8], [Ref. 9], [Ref. 10], [Ref. 11]. Their focus has been on search strategies, evaluating the impact of user training and investigating methods to address multiple logon protocols and retrieval language problems. However, based on these reports, user activity can be viewed from a different perspective. That is, while the user is connected to this interactive information resource, his activity may be categorized as either interactive or batch. Included as interactive are those user functions for search, refinements of search statements, perusal of hits, and if the database management system permits, narrowing down the hit-list during perusal. Some studies have described this kind of activity as cycles within cycles [Ref. 7] or a series of sifting [Ref. 12] through the body of retrieved data until the user is fairly satisfied with his 'find'. As stated earlier, there have been numerous investigations into this aspect of how the user spends his productive search time at the network resource. For completeness purposes, in describing the user's total connect-time, the studies mention users commands for hard-copy output. In the commercial database systems, where the charge is primarily

according to connect-time, there is a relatively small amount of time devoted to on-line printout. However, there is an indication that there may be a fair amount of off-line printing done at the database site (for a fee), and that output is mailed to the customer. We define batch work to be request for hard-copy output to the user site. As early as 1973, information scientists had been calling for an interface between batch and interactive in several areas, one of which is the transmission of large amounts of retrieved data to be passed later via batch [Ref. 13].

All network access to the commercial information resources are at dial-up speeds ranging from 300 to 2400 baud. All customer access, with a few exceptions which we will describe shortly, are through a variety of dumb dial-up terminals. The exceptions are found in non-Defense government agencies and at research institutions. The impetus of these efforts was to address the front-end problems of accessing a variety of retrieval systems each with their own logon protocols, query languages, and search strategies. The work in this area has been the development of an intelligent user interface [Ref. 14], [Ref. 15], [Ref. 16]. Their aim has been to aid the user in formulating search statements, and in some cases, because of its knowledge of the kinds of information in the network, the intelligent interface will attempt connections at all hosts with the database of interest until a connection is established. Early work in this area was the development of COnnector for Networked Information Transfer (CONIT) by Marcus at M.I.T. [Ref. 17]. Subsequent research using software based on CONIT has been carried on in Meadow's Individualized Instruction for Data Access (IIDA) [Ref. 15], [Ref. 18]. The Network Access Machine (NAM) was developed at the U.S. National Bureau of Standards (NBS) and Chemical Substances Information Network (CSIN) is now in place supporting the activities of the Environmental Protection Agency (EPA).

Although the study done by DImperio on SOLIS concentrates on user query habits, there is still division of customer work into interactive and batch. Only brief mention is made in the study to hard-copy requests. Although SOLIS has a large number of directly connected user terminals, this thesis is interested in looking at the network access and utilization of SOLIS. This is a database of reports, with average report size of approximately 2000 characters. Several years ago, the American Chemical Society (ACS) together with Bibliographic Retrieval Services, Inc. (BRS), embarked on a series of experiments to determine the usefulness of a full-text database (ACS Journal articles) and their availability for searching online. The experiment was based on a relatively small test database [Ref. 19], and looked only at the usefulness issue. No mention is made of output demands. However, we do not feel this in any way invalidates our perception of customer activity with a database. When ACS and BRS move to subsequent phases of the study with large test databases, we will probably then see their reports referring to print and display commands.

B. MODEL DESCRIPTION

To provide a general framework for describing the current approach and the proposed alternate, some elementary concepts from queueing theory are used. SOLIS is a facility with 15 servers. Each server provides the same service at identical rates. Customers arrive at the service facility from the network at a certain rate. If customer arrival follows a Poisson process and the service time is exponential with parameter mu, this defines an M/M/15 queueing system. In this simple example, all customers are collected into one source. CCINS has four sources, one from each of

the TASs. Figure 6.1 and Figure 6.2 represent the transition from one population pool to four separate pools.

Statistical analysis and tests for the customer arrival process to the TASs, found in Appendix A, show customers arrive at the TASs according to a Poisson process with parameter lambda(i), i = 1, 2, 3, 4. Arrival rates per hour

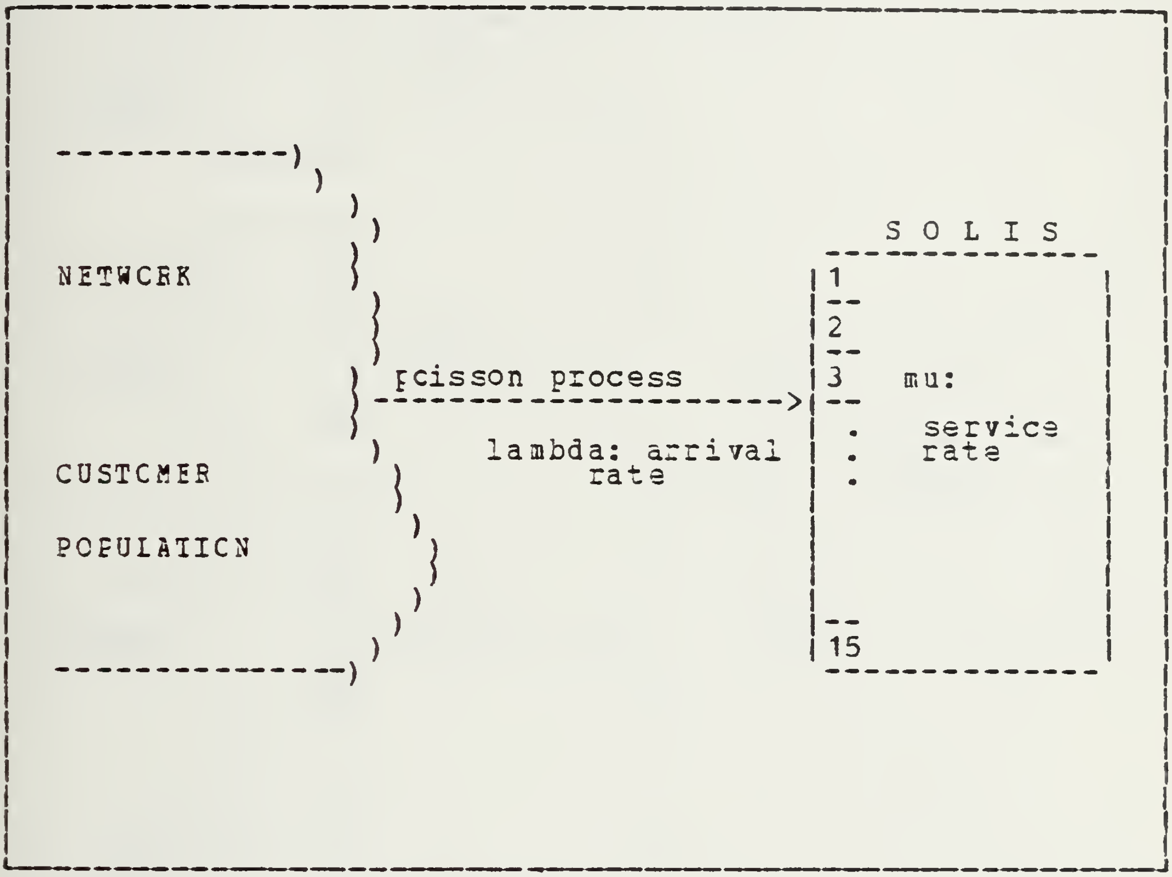


Figure 6.1 One population source to SOLIS.

at each of the TASs are 17.68, 9.19, 5.10 and 2.53. TASs are service facilities, offering network access, through their 24 ports. Once at a TAS, only a certain percentage of customers request service of SOLIS. The percentage of users requesting SOLIS access at each TAS is 0.70, 0.31, 0.098, and 0.89. A customer request is granted only if there is a

```
+-------
|      1 |
  --->|  TAS  1 |  percentage to SOLIS
17.68 |      -- |
      |       . |  0.70
cust./|       . |-------------
 hour |      -- |               |
      |      24 |               |
      +---------+               |
                                |
      +---------+               |
      |      1 |                |
  --->|  TAS  2 |               |
 9.19 |      -- |               |
      |   2   . |  0.31         |      +---------------
cust./|       . |---------------|      |1              |
 hour |      -- |               |      |--    SOLIS    |
      |      24 |               |      |2              |
      +---------+               |      |-- interactive |
                                |      |               |
      +---------+               |----->| .   resource  |
      |      1 |                |      | .             |
  --->|  TAS  2 |               |      | .             |
 5.10 |      -- |               |      |               |
      |   3   . |  0.098        |      |--             |
cust./|       . |---------------|      |15             |
 hcur |      -- |               |      +---------------
      |      24 |               |
      +---------+               |
                                |
      +---------+               |
      |      1 |                |
  --->|  TAS  2 |               |
 2.63 |      -- |               |
      |   4   . |  0.89         |
cust./|       . |--------------
 hcur |      -- |
      |      24 |
      +-------
```

Figure 6.2 Four population sources to SOLIS.

free network server on his local TAS and a free server at SOLIS. The user at SOLIS may be engaged in only interactive work during a session or may do both interactive and batch work in the session. SOLIS can now be described as a 2-stage facility, where all customers enter stage 1 and work there for a certin amount of time. They then proceed to stage 2 with a certain probability p or leave the system with probability 1-p. Figure 6.3 illustrates this concept. New customers can enter the server only if both stages are empty. From the SOLIS monitoring logs, 0.24 of the customers do only interactive work and 0.76 do both. The distribution of work-time for only interactive work is exponential with an average service time = 10.13 minutes. Similarly, the work-time distribution for both interactive and batch is also exponential with an average service time = 12.92 minutes. For the purposes of the analysis, the second work-time was separated into the individual times for interactive and batch. Statistical tests on the data showed that these distributions are also exponential with average service time for interactive work = 6.2 minutes and average service time for sending retrieved data = 6.8 minutes. This analysis may be found in Appendix A.

C. TIME-LINE INSPECTION

With the definitions of customer service-time from above, we now suggest a time-diagram which permits a graphical view of the partitions of work-time. Figure 6.4 is one such diagram. The begin and end times of a customer session are indicated. This session time is subdivided into interactive and batch parts. Considering the amount of time the interactive resource is occupied servicing this customer, then the proportion of interactive use is at the 0.6 level or 60% of the time. Another very simple situation is shown

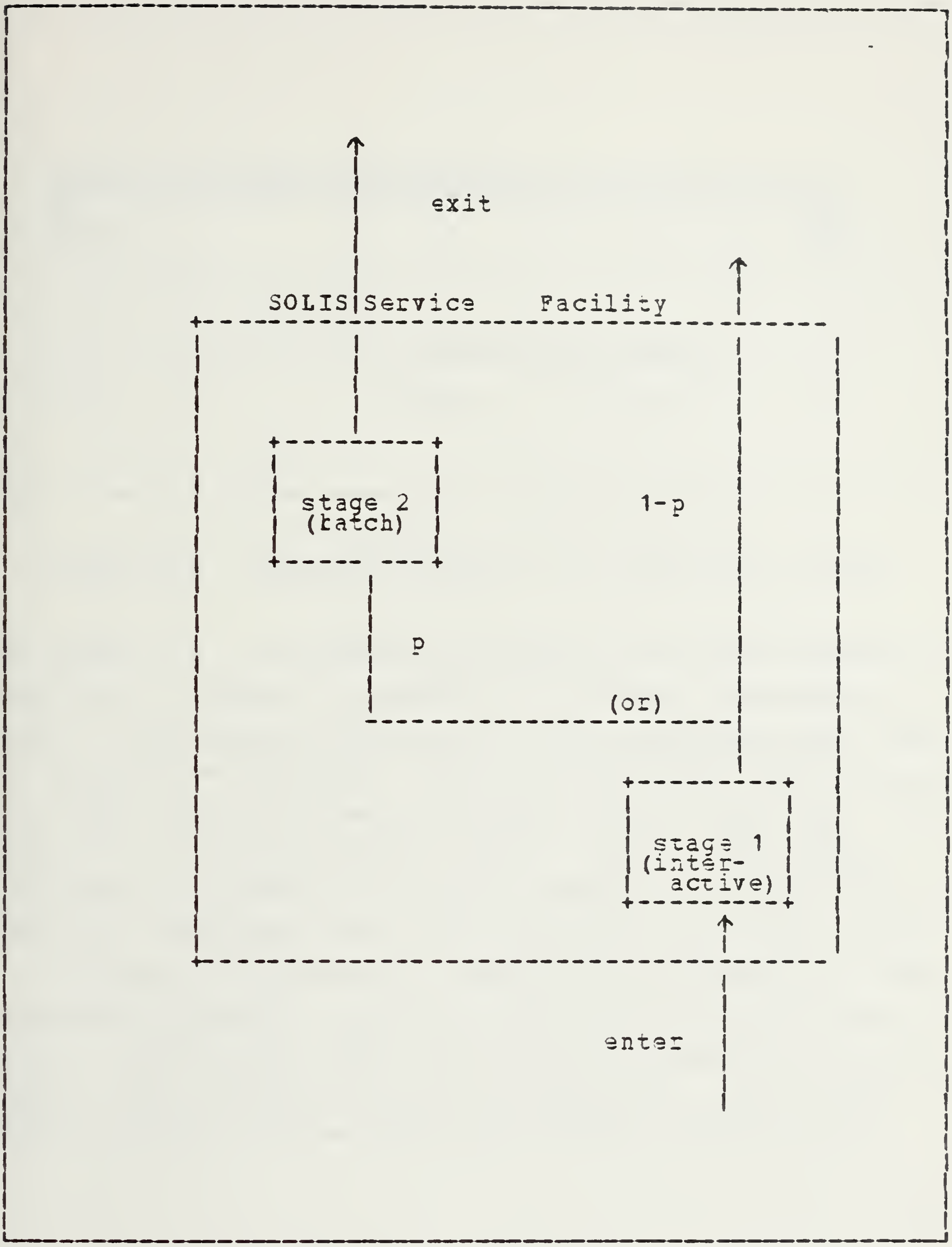


Figure 6.3 2-Stage service facility.

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX_____________________
+----+-----+----+----+----+----+----+----+----+----+-
0                        5                        10
begin                                             end


                       interactive: XXXXXX

                       batch      : ______
```

Figure 6.4 Time-Line diagram of 1 retrieval session.

in Figure 6.5 . The three horizontal time lines reflect the facility's ability to handle up to three customers at a time. Time again is divided into its respective work modes of interactive and batch. This case is simplified by having all three users starting and ending at the same time. The proportion of interactive use is 0.40. The number of customers the system was able to service in these ten time units is three and their average service time is ten units.

The real world does not operate in this manner. These diagrams were used to give some insight into the problem at hand. The computer simulation has incorporated the multiple IASs, their customer arrival distributions, percentages of SOLIS requests and the SOLIS work profile distributions.

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX_____________________
+----+----+----+----+----+----+----+----+----+----+-
0                        5                       10

XXXXXXXXXX_________________________________________
+----+----+----+----+----+----+----+----+----+----+-
0                        5                       10

XXXXXXXXXXXXXXXXXXXX_______________________________
+----+----+----+----+----+----+----+----+----+----+-
0                        5                       10
begin                                            end


                      interactive: XXXXXX
                      batch      : ______
```

Figure 6.5 Time-line of 3 retrieval sessions.

VII. AN ALTERNATIVE APPROACH:COUPLING OF INTERACTIVE AND BATCH

A major drawback of the current method is that SOLIS is set up as a 2-stage facility. Stage 2 is purely batch in nature and the server is kept busy transmitting data at a comparatively slow speed to what the server believes to be a terminal. While the customer is in stage 2, no new customer can enter the server to begin his stage 1 processing. With a substantial population of users, most of whom request a fair amount of data to be transmitted back to their TAS, either for printing at their local terminals or for further manipulation at their local TAS, we can easily foresee some problems. One way of addressing this kind of situation is to provide a high-speed background data-transmission facility between the server-cpu and user-cpu. Referring back to our model of SOLIS, we propose a transformation from a single node, 2-stage facility, to a 2-node tandem network as shown in Figure 7.1 Each box in that figure describes a queueing system consisting of a queue and server(s). Within each box is given the node number. Node 1 represents the interactive facility with 14 servers and node 2 is the batch facility with 1-server. The original 15 interactive ports on SOLIS are re-allocated to 14 interactive and one batch. When a customer has completed work in node 1 and has generated data to be sent back to him, his work request is forwarded to node 2 for processing. This arrangement now leaves node 1 free for interactive work.

The price for this design is that we usurp one of the 24 servers on TAS and one of the 15 servers on SOLIS. Using our simplified time-line diagram again, Figure 7.2 illustrates how three users can be accommodated on two

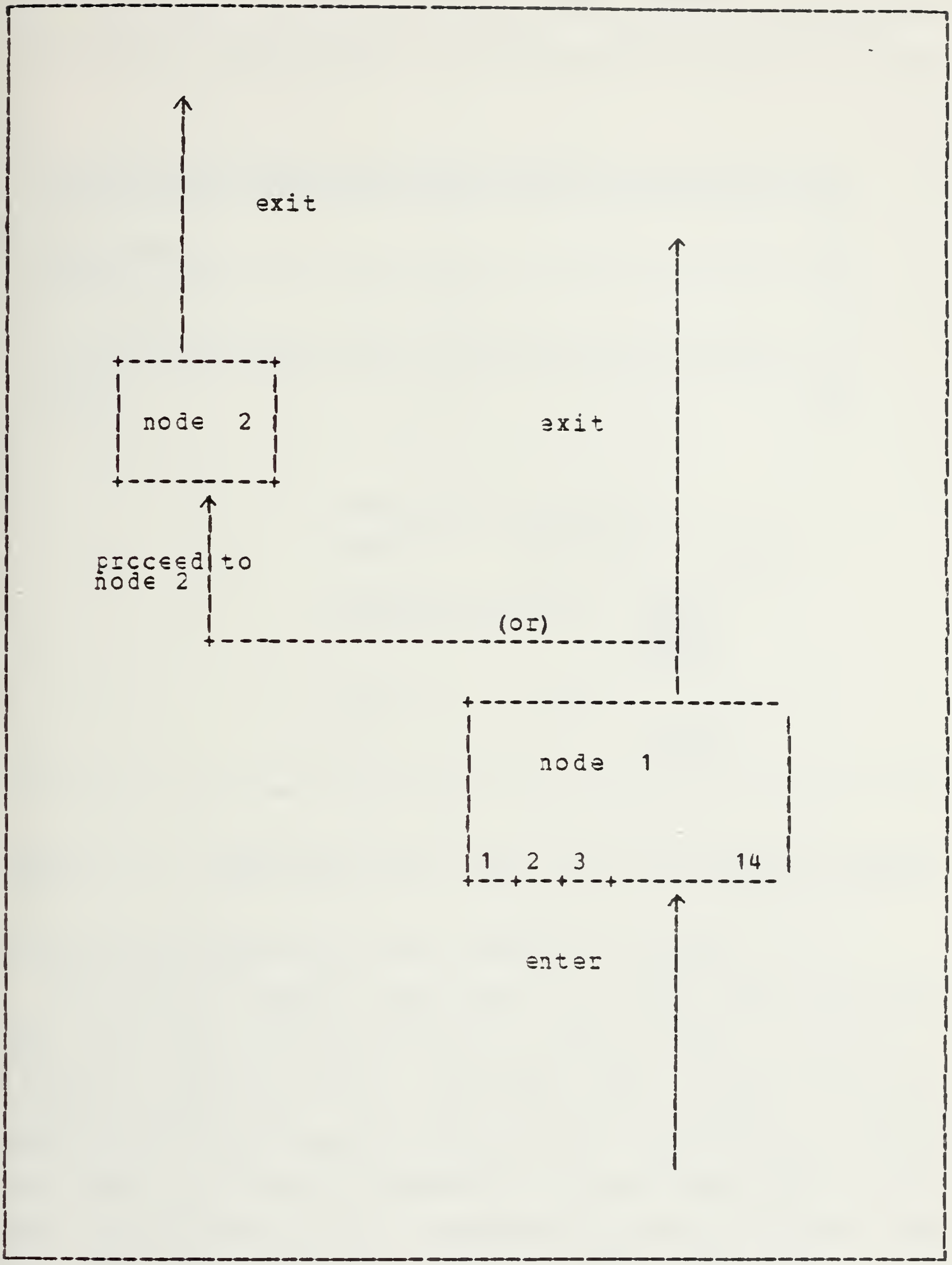


Figure 7.1 SCLIS as a 2-node tandem network.

interactive paths and all their data transfer requests on

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXYYYYYYYYYYYYYYYYYYY
+----+----+----+----+----+----+----+----+----+----+-
0                        5                        10

ZZZZZZZZZZ
+----+----+----+----+----+----+----+----+----+----+-
0                        5                        10

::******...
+----+----+----+----+----+----+----+----+----+----+-
--------------------------------------------------
0                        5                        10
begin                                            end


                   interactive resource: --------

                   batch       resource: --------
                                         --------

                   interactive work:  XXXXX
                                      YYYYY
                                      ZZZZZ

                   batch       work:   .....
                                       *****
                                       :::::
```

Figure 7.2 Time-line with 1 data and 2 interactive paths.

the one high-speed link with capacity to spare.

The other expense would occur in those cases when the user wants the data printed at his terminal as soon as possible. In the current situation, when the user gives the command to print the data, the system then begins to deliver the data at his terminal-printer at an approximate rate of 2400 baud. In this new proposal, the user must wait for the cpu to cpu transfer (at an approximate rate of 20 KB/s) and any queue time at node 2 before seeing any data at his printer. This particular side-effect must be considered

very carefully in the evaluation because this may be too much of an inconvenience to the customer.

VIII. THE NETWORK SERVICES MODEL

This chapter will discuss the model's goals, functional design, internal design, preliminary results, validation and applications.

A. DESIGN GOALS AND CONSIDERATIONS

The Network Services Model is a discrete event simulation that models network resource allocation in response to arriving customers' requests. It does not attempt to model the internal operations of the server hosts and user cpus, nor the particular flow of messages and packets through the COINS-II subnet. Instead, it focuses on modeling the network from the point-of-view of how the network capacity of the server and the user cpus is consumed in support of interactive database query and retrieval services. It simulates customer arrivals at the user cpus, their request for interactive network access, the allocation of network resources if available for the session, and their de-allocation at session completion. Design and implementation were motivated by the following seven design goals.

1. The model should be a faithful representation of the network entities contributing to interactive services.
2. The model should serve as a realistic simulation of customer activity and allocation of resources based on this activity.
3. It should be able to take the same customer activity and allocate resources based on the alternate proposal.
4. It should provide metrics for performance comparison between the two approaches.

5. It should serve as a measurement tool for doing sensitivity analysis as arrival rates and work profile distributions change.
6. As we acquire further insight into customer work habits (substantiated by more detailed logging and monitoring data), the model should be flexible enough to accommodate these statistics.
7. The model should be extensible and easy to modify so that it can serve as a long term design tool for the COINS/PMO.

One of the most important objectives of the model is that it be a faithful portrayal of the allocation of network capacity by the TASs, server-TASs and HOST when providing interactive network services. It is essential that the model behaves in a manner consistent with the flow of interactive job requests through COINS-II. Interactive job requests are initiated from a TAS or server-TAS, never from the HOST. For example, customers arrive at one of the TASs or server-TASs according to some distribution. Having arrived at a TAS, the customer then requests services to one of the interactive resource in the network. Customers at any TAS may request services at any of the server-TASs or HOST. On the other hand, customers at any server-TAS may request services at the HOST or any of the server-TASs except its own. If there is an available port at the user-cpu and the database-cpu, the demand is honored and the appropriate resources are allocated for the duration of the session. These events should occur in the model in the same fashion that they do in real life.

Statistics concerning customer arrival and their network requests were gathered from the accounting logs of the user-cpus. The model, therefore uses these distributions for the generation of network events. Characteristics of the session are another important aspect of this study. The

empirical data collected from COINS-II offers statistically sufficient information only for the SOLIS database host. Hence our particular analysis will be confined to evaluating network utilization of one database resource. It must be pointed out that this is not a limitation of the simulation model. It will support up to n-database resources, with the size of n dependent only on the size and capabilities of the computer the simulation model is run on. The description of this simulation will be of its full capabilites.

With the main issue being the comparison of two methods, there are two approaches to model implementation. One would be to implement two simulations, each reflecting a particular strategy; or implement one simulation using appropriate flag setting to regulate the simulation control flow for one strategy or other. Since only one aspect of interactive processing is changed, the later method is used. It was felt this is better than having to contend with maintaining two separate programs.

Since the model's main purpose is to furnish performance measurements of the two approaches, it must be able to take the same set of distributions and work profiles and execute for the current system and then execute for the alternate strategy. These distributions and work profiles of the database-cpus are input parameters to the simulation, thereby giving the model some level of flexibility. Performance measurements are done in the two general areas of number of customers refused and average system time. The model can be run with the provision of queues for customers awaiting network access. When the model is run with queues, further measurements are taken for the average wait time and the average wait time, given the queue is not empty. With respect to the alternative proposal, average system time is measured on two levels. One measure incorporates the time it takes to get all the output printed at the customer

terminal, (service-time1) while the other measure incorporates the time it takes to get all the output only to the user-cpu (service-time2). We feel it is important to make this kind of distinction because of the variety of intended uses of the retrieved data once it arrives at the user-cpu. The simplest activity is the mere printing of the data at the user-site. However, as reviewed in the literature and in light of the on-going work by the COINS/PMO, there is a definite shift from straight printing to some data massaging and some early efforts in data fusion. There are no machine logs available to indicate to what degree this is occurring, so these two statistics are computed to provide the range of possible expected system times. System-time2 is important also because it provides an indication of how much sooner a network path becomes free for re-use.

To be a practical design tool, the model should be able to be used by the COINS/PMO and its personnel to investigate the impact on customer services as the network grows with respect to more user-cpus and more interactive database-cpus. The analysis in this thesis is based on the current configuration and workload in COINS-II. However, in the next 12-month period, COINS expects to introduce two more TASs into the network and two more in the next two years. The model should be able to accommodate such changes in network configuration and workload. For this reason, information concerning each database-cpu and user-cpu are specified as run-time parameters to the model. The number of HOSTs, TASs and server-TASs and their respective profiles, including their network capacity and customer arrival rates are part of the data read in at run-time. This flexibility to adapt to network changes was a major influence for modular implementation.

SIMSCRIPT II.5 was used because it is equipped with the mechanics for handling discrete-event simulations and has much of the versatility of a general programming language. It has the traditional concepts of permanent and temporary entities, ownership and membership in sets and queues. To add a new TAS to the model, a minimum of three changes must be made to the simulation. A new ARRIVAL-event must be added to the structure, a command to initiate the start of arrivals for this new TAS, and finally, at the close of the simulation, there must be a command to terminate the arrivals for this new TAS. If the interarrival distribution of the customers to this new TAS is the same as one of the existing TASs, but with different arrival rate, then the same call can be made, using the different parameter. If however, the distribution is different, it is only a matter of writing a routine describing the distribution and calling this new routine for the next arrival. For the database-cpus, the only new work that would be required is when its work-profile distribution is different from any of the others already in the model. If it is different, a similar procedure must be followed as was described above for a new TAS.

B. HARDWARE AND SOFTWARE ENVIRONMENT

The Interactive Services Model runs on the IBM 3033 Attached Processor System under OS/VS2 at the W.R. Church Computer Center, Naval Postgraduate School. The software is a SIMSCRIPT II.5 program which has approximately 1,350 executable source language statements. Work areas are dynamically created during execution, depending on the input parameters. Attention must be given to the Job Control Language (JCL) setup with respect to execution time and storage requirements for the job. Runs for this thesis were

defined as a CLASS G job, permitting 15 minutes cpu time. The source code is the property of the U.S. government. Anyone interested in possible use of the program should contact the author. Operating instructions for the program can be found in Appendix B to this thesis.

C. FUNCTIONAL DESIGN

The model specifies when certain events are to occur, based on the distributions given for each event. In the alternate method, the queueing discipline of the high-speed facility contributes in deciding when departure events are to take place. The SIMSCRIPT II.5 timing routine actually handles the clock advances and the firing off of event processing according to schedule.

1. Customer Arrival

After initializing its internal tables based on input parameters, a customer arrival is scheduled for each of the user-cpus, and the simulation begins. There are a total of 13 events that can take place in a simulation run. Five events handle the arrival of customers to each of the user-cpus, and one terminates the simulation. The start state of the model is defined as no customers and no interactive network capacity is being used. Parameter input specifies the available resources for interactive services for each of the cpus. The remaining seven events are concerned with the sequence of events that begins with the customer requesting an interactive network service to the issuance of a command to initiate the retrieved data transfer and the steps involved in executing that command to the user-cpu, then to the database-cpu and the subsequent transmission of the data back through the network to the user terminal. The section on event handling provides a

more detailed description of these internally generated events. Each of the user-cpus have their own arrival rates whose inter-arrival times follows some distribution. The kind of distribution and its parameters can be specified as input parameters. Analysis of the inter-arrival times from the empirical data shows that its distribution is exponential. The SIMSCRIPT II.5 statistical distribution packages offers a fairly wide range of distribution functions to choose from. They include erlang, gamma and beta to mention a few. For a comprehensive list the reader should see [Ref. 20].

2. Resource Selection

Selection of which interactive database cpu is also derived from the empirical data. Throughout the simulation, the model maintains the currently available network capacity for each component. If there is sufficient network capacity at the user-cpu and the requested database-cpu, the appropriate resources will be busy for the duration of the session. If there is insufficient or no facility free, the request will either be refused or placed in a queue, depending on the run-time parameters. Each time resources become free, the earliest job request in a queue matching the available capacity is selected for processing.

3. Service Profile

Log data from each of the database cpus were used to define the session profiles of interactive service times, data transfer times and the percentage of customers requesting data transfers. These parameters can be modified at run time without changing the program code. Two possible things can occur at this point. If the customer has only interactive work to perform, the resources are tied up for just this period of time. However, should there be demand

for print-data, then the network resources are kept busy for the duration of the interactive portion plus the data transfer portion. The time defined for the data transfer portion is based on the distribution of output character size, transformed to number of bits divided by speed of the terminal. For example, if the number of characters is 50,000 and each character is transmitted as an 11-bit code with the terminal speed as 2400 baud, then the time for the transfer to take place is

(50,000 x 11) / 2400 = 229 seconds = 3.8 minutes

For the alternate proposal, when print-data is demanded, the interactive resource is freed for further interactive work, and the data is sent on the high-speed facility if it is free. If the high-speed facility is not free, the transfer request is placed in a queue until such time as the resource becomes available. For the purposes of this model, the network potential is estimated to be 20 KB/s. In the example given above, the transfer would take

(50,000 x 11) / 20,000 = 27.5 seconds = .45 minutes

This specific implementation permits three different distributions which are uniform, normal and exponential. This was done to indicate to potential users of the model, that the model is not restricted to only the exponential distribution, and that adding a new distribution is a simple exercise because the program only calls the statistical distribution functions of SIMSCRIPT.

Although this thesis is primarily concerned with how to get more interactive work done on the interactive resource, it must still consider how long it will take for the customer to eventually get his product. In the scenario just described, the data will be at the user-cpu in 27.8 seconds; however, it is not at the user terminal. And furthermore, that 27.5 seconds is straight transit time and does not include any queue time if the transfer request had

to wait in a queue. Should the customer want the data at his terminal, it would take another 3.8 minutes for it to be transferred from the user-cpu to the terminal. It is important that the model take these issues into consideration by keeping statistics on these different system times so that a fairer comparison can be made.

4. High-Speed Queueing Discipline

In the real system, when the network resources are busy, customer requests for SOLIS access do not wait in the queue; they are lost to the system. For the alternate method, queues can develop at the high-speed server. Its queueing discipline is described next.

Arrivals to this one-server facility can come from four different population sources or TASs. When the task arrives and the server is free, the data is transmitted immediately to the appropriate TAS. However, if the server is busy when the task arrives, the task is placed in a queue of work destined for the TAS from which the work originated. Once the server has started transmitting data to a TAS, it will continue to so until the queue for that TAS has been emptied. For example, let there be three TASs, denoted by TAS1, TAS2 and TAS3. Furthermore, let there be two data files for each of the TASs that the high-speed facility must transfer. The server will begin work on the TAS-queue whose task arrived the earliest. In this example, let the task in TAS1-queue have the earliest arrival time. Then the server will begin data transmission to TAS1 first. When that is completed, it will proceed to the second task in the queue for TAS1. If a new task arrives for transmission to TAS1 before the server has completed servicing the first two transmissions, the server will proceed to work on task three after it has completed the first two transmissions. Say now that the server has completed task three and no new work has

arrived for transmission to TAS1, then the server will pick the next earliest task waiting in the TAS2-queue and the TAS3-queue.

D. INTERNAL DESIGN

1. Overall Structure

Using SIMSCRIPT II.5 has given the INS model a very simple control structure. The permanent and temporary entities and their relationships to each other are defined in the PREAMBLE section of the model. All the events, their attributes and priority handling are also declared in this section. The global variables and any counting and averaging are specified here. Program MAIN is concerned only with managing the general flow of control including the setting up of the initial system state and providing the starting events that will set the simulation in motion. The simulation is not begun until an explicit statement START SIMULATION is issued by MAIN. At that point, control is transferred to the SIMSCRIPT II.5 timing mechanism. The timing mechanism manages the flow of control from event to event as they are scheduled to occur. When an event processing has completed, the timer searches through the events-list, looking for the earliest next-event to schedule. The timer then updates the system clock and transfers control to the event routine. When no further events are found on the events-list, the timer returns control back to MAIN.

To illustrate this point, the following example and accompanying seven Figures are provided. Let there be two TASs, denoted as TAS1 and TAS2. Suppose TAS1 has two network ports and TAS2 has three network ports, and let SOLIS have two network ports. Figure 8.1 is a picture at the start of the simulation, the clock is at time = 0

minutes. All the resources are free and are indicated by empty boxes. In Figure 8.2, the clock has advanced to time = 2 minutes and shows a customer has arrived and requested network access to SOLIS. The Figure shows a path established between TAS1 and SOLIS and the allocation of the network ports. The port boxes are Xed. Customer1 is in an interactive session with SOLIS. From customer work profile, it is determined that customer1 will have an interactive session of 6 minutes, followed by request for hard-copy that will last 7 minutes. According to the interarrival distribution of customers, customer2 arrives at TAS2 at time = 5 minutes, with a work profile of 6 minutes of interactive work and no hard-copy request. Figure 8.3 shows the state of the model at time = 5 minutes. There are now two paths to SOLIS, one from TAS1 that started at time = 2 minutes and the second one that started in this time. The next event to occur is at time = 8 minutes when customer1 at TAS1 completes his interactive work and now goes into batch processing. This is indicated in Figure 8.4 and the path doing batch processing is indicated as B. Another customer arrives at TAS1 is the next event that occurs at time = 10 minutes. Figure 8.5 shows an attempt to establish a path between TAS1 and SOLIS, but is not successful because of insufficient resource at SOLIS. Customer3 is lost to the system. The next event occurs at time = 11 minutes when customer2 has completed his interactive work with SOLIS. Figure 8.6 shows the path between TAS2 and SOLIS is now free. The next event occurs at time = 14 minutes when customer1 has completed his batch processing. Figure 8.7 now shows the path between TAS1 and SOLIS is now free.

```
 time = 0

TAS1                                          SOLIS
+--+                                          +--+-
|  |                                          |  |
+--+                                          +--+-
|  |                                          |  |
+--+                                          +--+-

TAS2
+--+-
|  |
+--+-                    Event 1 at time = 0
|  |
+--+-
|  |
+--+-
```

Figure 8.1 Initial State.

```
 time = 2

TAS1                                          SOLIS
+--+ customer 1                               +--+-
|XX|------------------------------------------|XX|
+--+                                          +--+-
|  |                                          |  |
+--+                                          +--+-

TAS2
+--+-
|  |
+--+-                    Event 2 at time = 2
|  |
+--+-                    Interactive session: -----
|  |                     Port occupied      : XX
+--+-
```

Figure 8.2 State 2.

```
 time = 5

TAS1                                                      SOLIS
+--+customer 1                                            +--+-
|XX|------------------------------------------------------|XX|
+--+                                                      +--+-
|  |                                                  ////|XX|
+--+                                             /        +--+-
                                            /
TAS2                                   /
+--+customer 2                     /
|XX|////////////
+--+                                Event 3 at time = 5 minutes
|  |
+--+-                               Interactive session: -----
|  |                                                   : /////
+--+-
                                    Port occupied      : XX
```

Figure 8.3 State 3.

```
 time = 8 minutes

TAS1                                                      SOLIS
+--+customer 1                                            +--+-
|XX|BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB|XX|
+--+                                                      +--+-
|  |                                                  ////|XX|
+--+                                             /        +--+-
                                            /
TAS2                                   /
+--+customer 2                     /
|XX|////////////
+--+                                Event 4 at time = 8 minutes
|  |
+--+-                               Interactive session: -----
|  |                                                   : /////
+--+-
                                    Batch work         : BBBBB
                                    Port occupied      : XX
```

Figure 8.4 State 4.

```
 time = 10 minutes

TAS1                                               SOLIS
+--+customer 1                                     +--+-
|XX|BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB|XX|
+--+                                               +--+-
|XX|customer 3, blocked          /        ////|XX|
+--+                         /                     +--+-
                         /
TAS2                 /
+--+customer 2   /
|XX|////////////
+--+                    Event 5 at time = 10 minutes
|  |
+--+-                   Interactive session: -----
|  |                                       : /////
+--+-
                        Batch work         : BBBBB
                        Port occupied      : XX
```

Figure 8.5 State 5.

```
 time = 11 minutes

TAS1                                               SOLIS
+--+customer 1                                     +--+-
|XX|BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB|XX|
+--+                                               +--+-
|  |                                               |  |
+--+                                               +--+-

TAS2
+--+-
|  |
+--+                    Event 6 at time = 11 minutes
|  |
+--+-                   Interactive session: -----
|  |                    Batch work         : BBBBB
+--+-                   Port occupied      : XX
```

Figure 8.6 State 6.

```
 time = 14 minutes

TAS1                                                  SOLIS
+--+                                                  +--+-
|  |                                                  |  |
+--+                                                  +--+-
|  |                                                  |  |
+--+                                                  +--+-

TAS2
+--+-
|  |
+--+                          Event 7 at time = 14 minutes
|  |
+--+-                         Interactive session: -----
|  |                          Batch work         : BBBBB
+--+-                         Port occupied      : XX
```

Figure 8.7 State 7.

For each set of network configuration, four runs are made. The two major categories are the current and the alternate proposal. And within each of these cases, the model is run two times, one for the situation where customers go away when the network resources are not available; and the other, where they are placed in a queue. All queues are considered to be first-in-first-out (FIFO). Sequencing through these four iterations is managed by MAIN.

2. Event Handling

There are 13 events that can occur in the model. They will be described in chronological order. The reader is referred to Appendix F of this thesis for the logic diagrams of these internally generated events.

a. Events 1-5: TASx.ARRIVAL, where x = 1, 2, ..., 5

There are five events that handle the customer arrivals at each of the user-cpus. The work performed in these events are:

1) schedule the next arrival according to the distribution for this user-cpu.
2) determine which interactive network host to request services.
3) determine the work profile at this interactive host.
4) if the current network capacity permits, seize the appropriate resources.
5) if there is to be no print-data command, schedule a departure at the end of the interactive portion.
6) if there is to be print-data command, schedule the event to handle the print-data command.
7) If the current network capacity is not able to satisfy the request, file the request into the queue for new work or ignore the request, depending on the input parameter.
8) update the appropriate statistics gathering variables.

b. Event 6: THDEPART

The next event is the departure from the system at the end of the interactive portion. The work performed here are:

1) release the interactive network resources.
2) If there are any other departures of this same nature in the events-list that is to occur at this same time instant, process this event by releasing the interactive network resources used by this event.
3) Having updated the network availability, search through the queues of new work for any job request

that can be satisfied and schedule the appropriate events. If there are no job requests waiting in the queue, return to the main timing routine.

4) Update the appropriate statistics gathering variables.

c. Event 7: USEND

Event to handle sending print-command to the user-cpu. Schedules the event at the user-cpu to handle the print-command in the amount of time to send the command from the customer terminal to he user-cpu.

d. Event 8: UC.ARRIVAL

The event to process the print-command at the user-cpu performs the following:

1) If the user-cpu is busy handling another command, file the request in the queue for the user-cpu.
2) If the user-cpu is free, set the user-cpu flag to busy and schedule the event to release user-cpu resource in the amount of time to process the request.

e. Event 9: UC.DEPART

The event to free the user-cpu after processing the print-command does the following work:

1) Set the user-cpu flag free.
2) Schedule event to handle print-command at the server-cpu.
3) If there is more work in the user-cpu queue, take the next task, set the flag to busy and schedule the event to release user-cpu resource in the amount of time to process the request.
4) If the queue is empty, return to the main timing routine.

f. Event 10: SC.ARRIVAL

Event at the server-cpu to handle the print-command does the following work:

1) If server-cpu is busy handling another print-command, file the request in the queue for the server-cpu.
2) If the server-cpu is free, set the server-cpu flag to busy and schedule the event to release the server-cpu resource in the amount of time to prepare the data for transmission.

g. Event 11: SC.DEPART

Event to free the server-cpu after preparing the data for transmission performs the following work:

1) set the server-cpu flag free.
2) Schedule the event to send the data to the user.

Current method: Schedule release of interactive resource in the amount of time to transmit the data.

Alternate method: Schedule release of the interactive resource now and send the requested data on the high-speed facility if it is available, otherwise place it in the queue of work for the high-speed facility.

3) If there are pending print-command tasks in the queue, work on the earliest task, set the flag busy and schedule the event to release the server-cpu resource in the amount of time to prepare the data for transmission.
4) For the alternate proposal, release the interactive resource and look for other events that are completed at this same time.

h. Event 12: LTHDEPART

The event handles the departure from the system of those who requested data transfers. The appropriate statistics gathering variables are updated. Based on what departure has occurred, the appropriate network resources are released. However, for the alternate configuration, where we are using a high-speed facility to pass all print-data output, this particular resource is not released until its queue is emptied.

i. Event 13: CLOSING

This event cancels the scheduled TAS arrivals.

3. Data Structures

SIMSCRIPT II.5 provides a framework for handling concepts in simulation such as permanent and temporary entities, queues and events.

a. Permanent Entities

There are four kinds of permanent entities. The first two are HOST and TAS. The server-TAS is included as both a HOST and a TAS because it really serves these two functions. The important attributes for the HOST and TAS are their maximum number of network ports and a flag-field to denote when it is a server-TAS. Since the model handles in detail, the sequence of events beginning with the user issuing the print-command, additional attributes of a busy-flag and a queue have been defined.

To handle TAS to HOST connectivity, a permanent entity called TASHOST is defined. The important attributes of this entity are the maximum network paths between a given TAS and a given HOST and identification of this given TAS and given HOST. For example, suppose there are two TASs,

called TAS1 and TAS2 and one HOST. TAS1 has a capacity for ten interactive ports, TAS2 has a capacity for 25 interactive ports, and HOST has a capacity for 15 interactive ports. This results in two permanent entities called TASHOST. The first one is for connectivity between TAS1 and HOST with a maximum possible capacity of ten interactive paths. The second TASHOST is for the connectivity between TAS2 and HOST with a maximum possible capacity for 15 interactive paths.

The fourth type of permanent entity is same in concept as TASHOST and is called LPATH, reflecting the high-speed data transfer facility between a TAS and a HOST. LPATH has a queue and the attributes to identify which TAS and which HOST.

b. Temporary Entities

Temporary entities are created and destroyed during the course of the simulation and are called tasks. They are created only when a request for service cannot be honored because the service facility is busy. They are placed in a queue and removed only when a server becomes free. All queue disciplines are first-in-first-out (FIFO). There is a potential for four different kinds of temporary entities that can be created during a simulated run. They are:

TASK: Created when there is insufficient network capacity to support an interactive session. It is placed in the appropriate TASHOST queue.

UTASK: Created when the TAS is busy handling another user request to send the print data command to the HOST. It is placed in the queue for the TAS.

STASK: Created when the HOST is busy handling another user request to prepare data for transmission to the user. It is placed in the queue for the HOST.

LTASK: Created only in the model of the alternate proposal. It is created when the high-speed facility LPATH is busy servicing another transmission request. It is placed in the appropriate LPATH queue.

c. Parameters

To make the simulation as flexible as possible, the program has the mechanisms for describing the desired network configuration and characteristics at run-time.
HOST characteristics include:

1) number of interactive ports;
2) proportion of customers doing only interactive work;
3) for customers doing only interactive work, the distribution and its parameters which describe this service time; The simulation expects service time in minutes.
4) for requests of hard-copy output, the distribution and its parameters describing the amount of characters that is to be transmitted; The simulation expects the number of characters and will make the transformation into the amount of time to transmit the data.
5) for sessions where a user will do both interactive and batch work, the distribution of the interactive portion of the session and its parameters describing this service time. The simulation expects interactive service time in minutes.

TAS characteristics include:

1) arrival rate of customers to the TAS; The simulation expects this to be in the number of customers per hour. Furthermore, the simulation assumes this to be a Poisson process.
2) number of interactive ports;
3) of the customers arriving, the proportion which will request some network access;

4) for those TAS customers requesting network access, the proportion of users using each of the network database facilities;

E. PRELIMINARY RESULTS

There is particular interest in the comparison of performance as the arrival rates and the amount of data to be transferred at the end of an interactive session is increased. The methodology adopted was to run the model with the empirical data from COINS-II and establish that as the baseline. This baseline consisted of four TASs and one interactive host. All four TASs have 24 available ports for network access. Table I shows the parameters of the baseline configuration of the four TASs. Customer arrival rates to these TASs ranged from 2.63 to 17.68 customers per hour. The proportion of customers selecting to go to SOLIS ranged between 0.70 to 0.82. After establishing this baseline, several runs were made increasing only the arrival rates of

TABLE I

TAS operating characteristics

TAS #	# ports	arrival rate	% to SOLIS
TAS 1	24	17.68	0.70
TAS 2	24	9.19	0.31
TAS 3	24	5.10	0.098
TAS 4	24	2.63	0.82

(customers/hour)

TAS2 to TAS4 until they all reached 17.68 customers per hour. The next series of runs started with the four TASs and progressed up to eight TASs. All arrival rates were 17.68 customers per hour.

The other variable of interest is the amount of data to be transmitted on this interactive connection. The approach to this was similar to what was done with the arrival rates. Starting with the original empirical data, subsequent runs involved increases in the data transfer amount. The initial run was for 89,018 characters. This was increased by 10% increments to 209,900 characters.

The next series of runs involved the addition of new TASs to the network first starting with the original data transfer demands and proceeding up to 209,900 characters. In this third series of runs, each of the TASs had an arrival rate of 17.68 customers per hour.

Table II shows the results of the baseline configuration and workload as the arrival rates were varied. They do not show any problem with the expected customer loss. The times given are in minutes. In the alternate method, two kinds of system time were measured. System time1 includes the transmission of data to the user terminal, while system time2 considers the service completed as soon as the data is received at the TAS. As expected, system time1 is always larger than the system time for the current method. The differences range from 44 to 52 seconds. On the other hand, comparing system time2 with the system time of the current method, system time2 is smaller by about 4.5 minutes. (range of 4 minutes 30 seconds to 4 minutes and 42 seconds). Graphing these results in Figure 8.8 presents a better picture of where system time1 and system time2 lie in relation to the system time of the current method. With respect to user services, Figure 8.8 indicates that, if the customer wants the data printed at his terminal site immediately, he

TABLE II

Increasing arrival rates

CHANGE IN ARRIVAL	CURRENT SYSTEM TIME	CURRENT EXPTD LOSS	ALTERNATE SYSTEM TIME1	ALTERNATE SYSTEM TIME2	ALTERNATE EXPTD LOSS
1	12.43	0.0	13.17	8.01	0.0
2	12.39	0.0	13.17	8.01	0.0
3	12.41	0.0	13.18	8.00	0.0
4	12.42	0.0	13.21	8.03	0.0
5	12.43	0.0	13.23	8.03	0.0
6	12.38	0.0	13.19	8.01	0.0
7	12.35	0.01	13.17	8.02	0.0
8	12.36	0.0	13.20	8.04	0.0
9	12.35	0.004	13.21	8.05	0.0

ARRIVAL RATES

1: 17.68, 9.19, 5.10, 2.63 CUSTOMERS PER HOUR
2: 17.68, 17.68, 9.19, 5.10 CUSTOMERS PER HOUR
3: 17.68, 17.68, 10.11, 5.61 CUSTOMERS PER HOUR
4: 17.68, 17.68, 12.02, 6.20 CUSTOMERS PER HOUR
5: 17.68, 17.68, 14.80, 7.47 CUSTOMERS PER HOUR
6: 17.68, 17.68, 17.68, 9.04 CUSTOMERS PER HOUR
7: 17.68, 17.68, 17.68, 12.02 CUSTOMERS PER HOUR
8: 17.68, 17.68, 17.68, 14.80 CUSTOMERS PER HOUR
9: ALL 4 IAS'S AT 17.68 CUSTOMERS PER HOUR

times are in minutes

will have to wait about a minute longer in method 2 than he does currently. On the other hand, if the customer is not interested in printing the data immediately, or wants to have it merged with other query results at a later time, he can be completed with his work about 4.5 minutes sooner in method 2 than he does currently.

Table III shows the results of the baseline configuration and workload as the data amounts to be transferred was increased. System time1 exceeds the system time of method 1

```
MINUTES
13.6
    -                              B       B       B       B               B       B
    -        B       B                                             B
    -
    -        A       A       A       A       A       A       A       A       A
12.2
    -
    -
    -
    -
10.8
    -
    -
    -
    -
 9.4
    -
    -
    -
    -
 8.0+        C       C       C       C       C       C       C       C       C
     +---------------+---------------+---------------+---------------+----
    0.0             2.0             4.0             6.0             8.0     9.0
                                    ARRIVAL RATES

      A:   System time for current method
      B:   System time1 (to the terminal) for alternate
      C:   System time2 (to the cpu) for alternate

  1:   17.68, 9.19, 5.10, 2.63  CUSTOMERS PER HOUR
  2:   17.68, 17.68, 9.19, 5.10 CUSTOMERS PER HOUR
  3:   17.68, 17.68, 10.11, 5.61 CUSTOMERS PER HOUR
  4:   17.68, 17.68, 12.02, 6.20 CUSTOMERS PER HOUR
  5:   17.68, 17.68, 14.80, 7.47 CUSTOMERS PER HOUR
  6:   17.68, 17.68, 17.68, 9.04 CUSTOMERS PER HOUR
  7:   17.68, 17.68, 17.68, 12.02 CUSTOMERS PER HOUR
  8:   17.68, 17.68, 17.68, 14.80 CUSTOMERS PER HOUR
  9:   ALL 4 TAS'S AT 17.68 CUSTOMERS PER HOUR
```

Figure 8.8 System times with increasing arrival rates.

between 44 seconds to 2 minutes and 15 seconds, as the transfer size increases. The range of differences between system time2 and the system time of the current method ranges between 4 to 10 minutes, as the transfer size increases. Expected customer loss is again zero for both

TABLE III

Increasing data transfer sizes

# chars.	current system time	current exptd loss	alternate system time1	alternate system time2	alternate exptd loss
89018.	12.43	0.0	13.17	8.01	0.0
97920.	12.95	0.0	13.78	8.10	0.0
107712.	13.51	0.0	14.45	8.20	0.0
118483.	14.14	0.0	15.18	8.31	0.0
130331.	14.83	0.0	16.01	8.45	0.0
143364.	15.58	0.0	16.91	8.60	0.0
157701.	16.41	0.0	17.92	8.77	0.0
173471.	17.33	0.0	19.06	9.00	0.0
190818.	18.34	0.0	20.31	9.24	0.0
209900.	19.45	0.0	21.70	9.52	0.0

times are in minutes

cases. Graphing these results in figure 8.9 shows the relationships between these system times.

Based on the assumptions of the model, results thus far seem to suggest there is no apparent danger of customer loss either at the present workload or as the arrival rates and data transfer sizes are increased for the four TASs. The disadvantage of method 2 is the extra amount of time (between 1 to 2 minutes) the intelligence analyst must remain at the terminal to have his output printed immediately at his printer. If the customer does not require the printout immediately, there is an advantage because he completes his work somewhere between 4 to 10 minutes sooner. This also means there is an extra 4 to 10 minutes during which another interactive search and refinement session can be started. Considering data fusion efforts sponsored by the CCINS/PMO, this 4 to 10 minutes is an advantage to the TAS because it can receive the data in this shorter amount

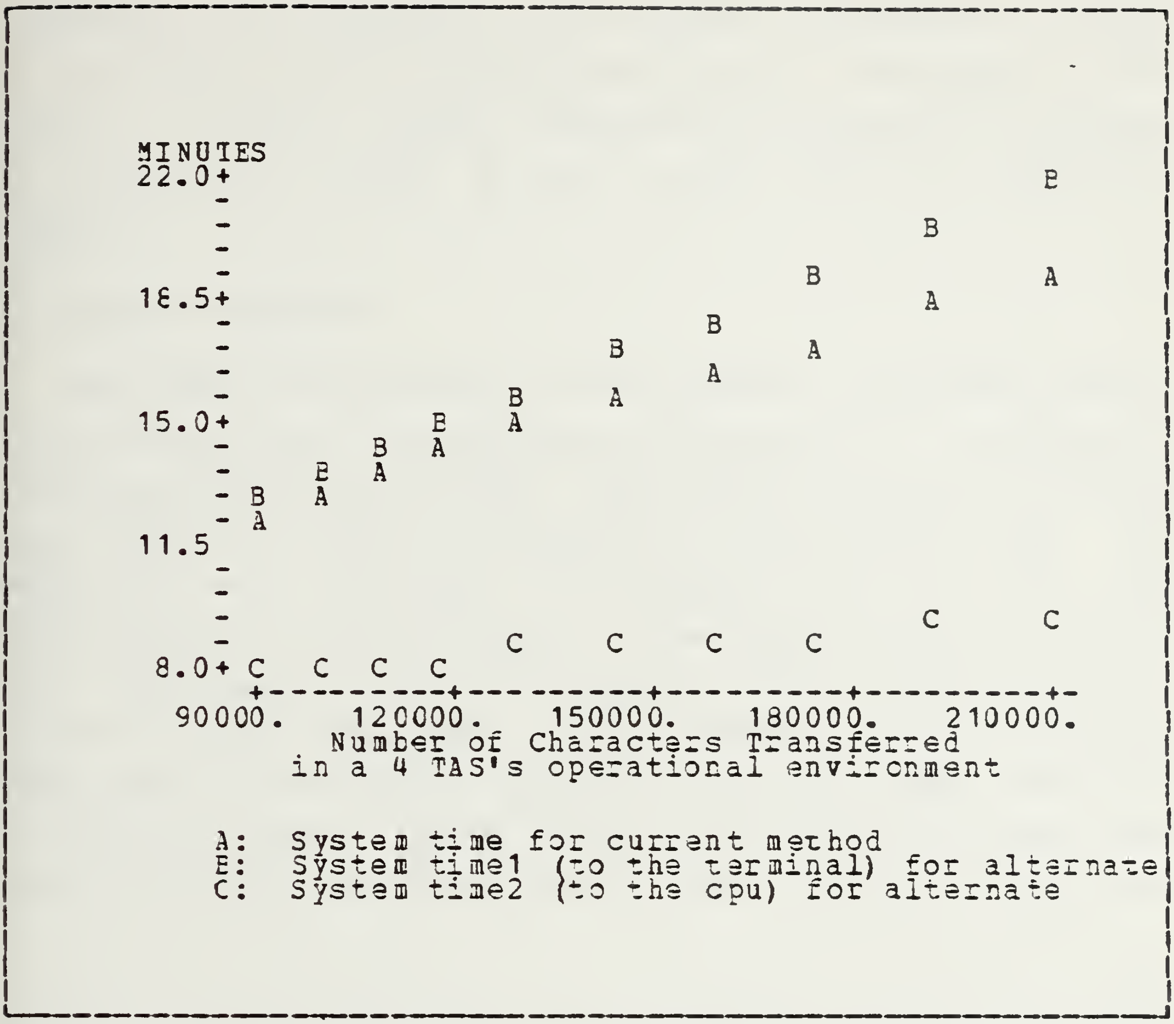


Figure 8.9 System times with increasing transfer sizes.

of time and can proceed with the work of data fusion that much sooner.

A valuable asset of any model is its ability to help us answer the "What if..." questions illuminating potential problems and benefits. They can aid us in determining some course of action in long-range systems planning. We have just looked at the cases where arrival rates and data transfer amounts were varied within the present COINS-II environment. Further examination of the changes of these

parameters are required as the network environment changes, in particular as more operational TASs are introduced to the network. The next chapter contains the comapartive analysis of these two methods as one to four TASs are added to COINS-II.

F. MODEL VALIDATION

The model is validated on the observed reference points. Its results from the baseline configuration is compared to the measurements from actual performance for average system time, customer loss and proportion of interactive use. The empirical data indicated no customer loss due to non-availability of network ports. The average system time for all SOLIS users was 12.19 minutes and the proportion of interactive use was 0.602. Using customer arrival rates and work profiles from the empirical data, the model predicted an expected system time of 12.43 minutes, no customer loss and 0.60 proportion of interactiv use. Table IV shows this

TABLE IV
Model validation results

	observed	model results
system time (minutes)	12.19	12.43
customer loss	0.0	0.0
proportion of interactive use	0.60	0.60

comparison. The important underlying assumptions of this model are the distributions describing customer inter-arrival times and customer service times. As long as the arrival process continues to be Poisson, with parameter lambda, and the service time remains exponential with parameter mu, then the results from the model may be considered valid.

G. MODEL APPLICATIONS

Our investigations are based on the demands of four TASs on SOLIS. Network demands on USIS and NUIS were not included because of the lack of empirical data describing those activities. However, the INS model has been designed and implemented to handle these kinds of network services. As soon as COINS can collect such customer profile information, it can simply be given to the model as input parameters. No program modification is required.

Although the discussion of model entities were in terms of HOSTs and TASs and server-TASs, the reader should be reminded of their definitions in order to find a more general application of the INS model. HOSTs are pure servers, and TASs are pure users. Server-TASs, on the other hand, are a hybrid user and server which both offer services to and uses services from the network. Hence, when a new node is added to COINS, it can be categorized as a server, user, or hybrid. For example, when a gateway between COINS and some network X is installed, it too can be classified as one of the three entities. If the gateway provides two-way service of permitting users in network X to access COINS services and permitting COINS customers to access services in X, then the gateway may be called a hybrid system. It can be incorporated into the model as a server-TAS.

IX. COMPARATIVE ANALYSIS

The COINS/PMO is actively involved with the installation of three new TASs. One TAS will be located at Lawrence Livermore Laboratory in California, and the second one will be at the State Department in Washington, D.C. A variation of the basic TAS will be installed at DIA and serve as a gateway between IDHSC and COINS, permitting IDHSC customers interactive access to COINS. For the purposes of our analysis, this gateway is a TAS. It is a source of interactive customers to COINS-II. The COINS/PMO is engaged in preliminary discussions with several intelligence organizations for installations of a TAS at their sites. In light of this customer growth over the next several years, we feel it would be useful to ask the model the "What if ..." questions. What if we had five operational TASs? ...six operational TASs? and on up to eight operational TASs.

The methodology was to run the model, establishing a baseline of four fully operational TASs. Then for each subsequent run, add a TAS functioning in an operational mode. The model was iterated five times, starting with the configuration of four TASs and ending with a total of eight TASs. Table V summarizes the results of these runs. Once again, system time1 exceeds system time of method 1 by about 1 minute, and system time2 is less than system time of the current method by about 4 minutes. Figure 9.1 is the graph of these results. The interesting result is expected customer loss. There is a jump from a 0% expected loss with four operatonal TASs to a 4% expected loss with five operational TASs in the current method. The alternative is still at a zero expected loss. For method 1, once there is a non-zero expected loss at five TASs, expected loss increases about 8.6 percentage points each time a new TAS is added,

TABLE V

Adding a TAS to the network

# TAS's	current system time	current exptd loss	alternate system time1	alternate system time2	alternate exptd loss
4	12.35	C.000	13.21	8.050	0.0
5	12.33	0.039	13.24	8.110	0.0
6	12.24	0.115	13.23	8.130	0.010
7	12.30	C.215	13.29	8.190	0.040
8	12.30	C.300	13.35	8.220	0.089

times are in minutes

ending with a customer loss rate of 30% for eight TASs. On the other hand, in method 2, the first non-zero expected customer loss is at six TASs, with about a 4 percentage point increase for each additional TAS, ending with a customer loss rate of 9% for eight TASs. The arrival rate of 17.68 customers per hour was used for all TASs. These expected loss values are plotted in Figure 9.2 . The slope of the line describing expected loss for the current system is 0.0768 while the slope for the alternative system is 0.0217.

Reviewing these findings in light of the third measure of proportion of interactive work will provide information on what proportion of the session time is consumed in doing interactive work. Using the same runs to construct Table V for the comparisons of system times, Table VI was constructed showing the proportion of interactive use as a new TAS is added to each iteration. It can be seen immediately that for method 1, the proportion of interactive use is only at 0.61 while expected customer loss rises to 30%.

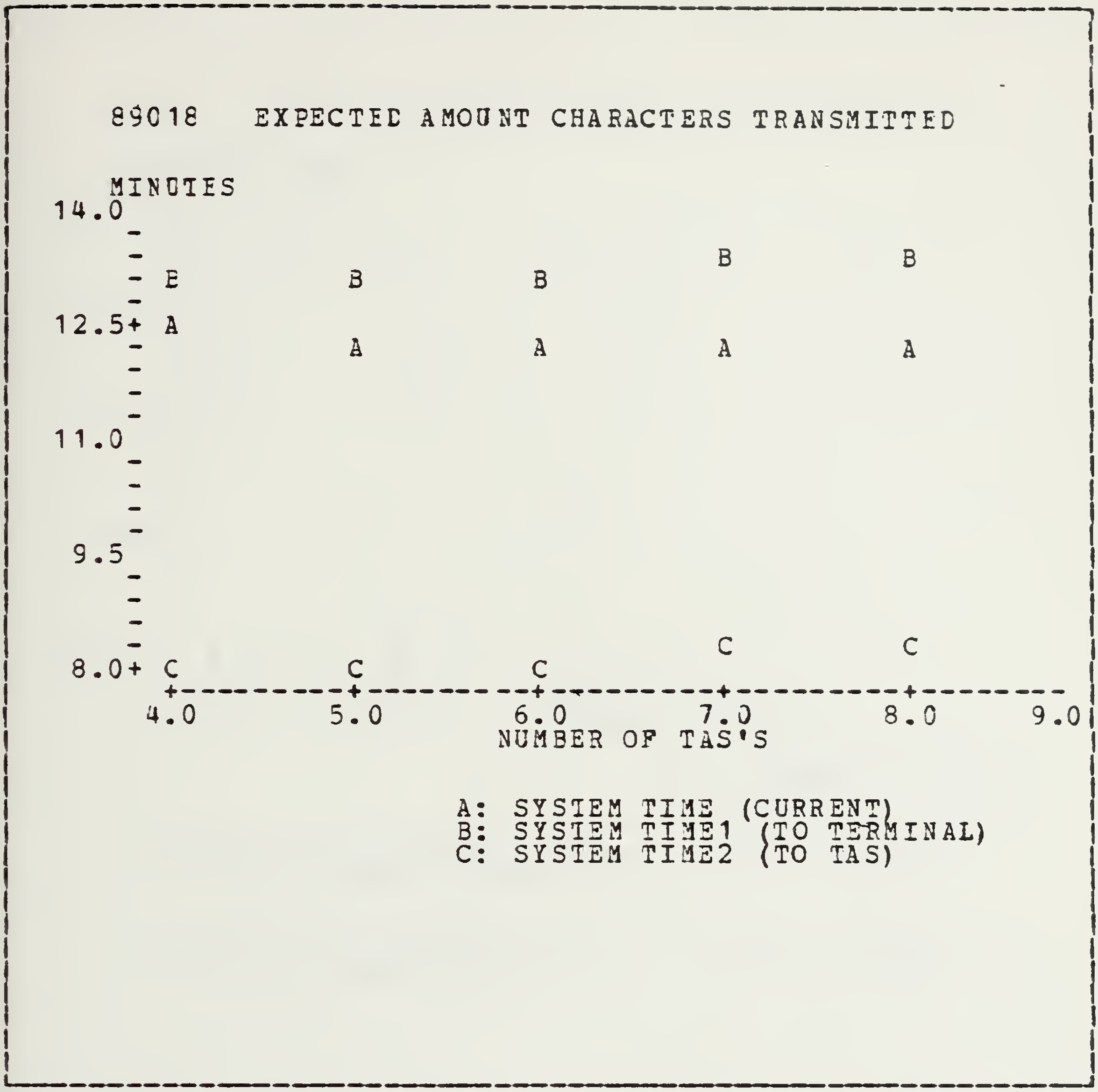
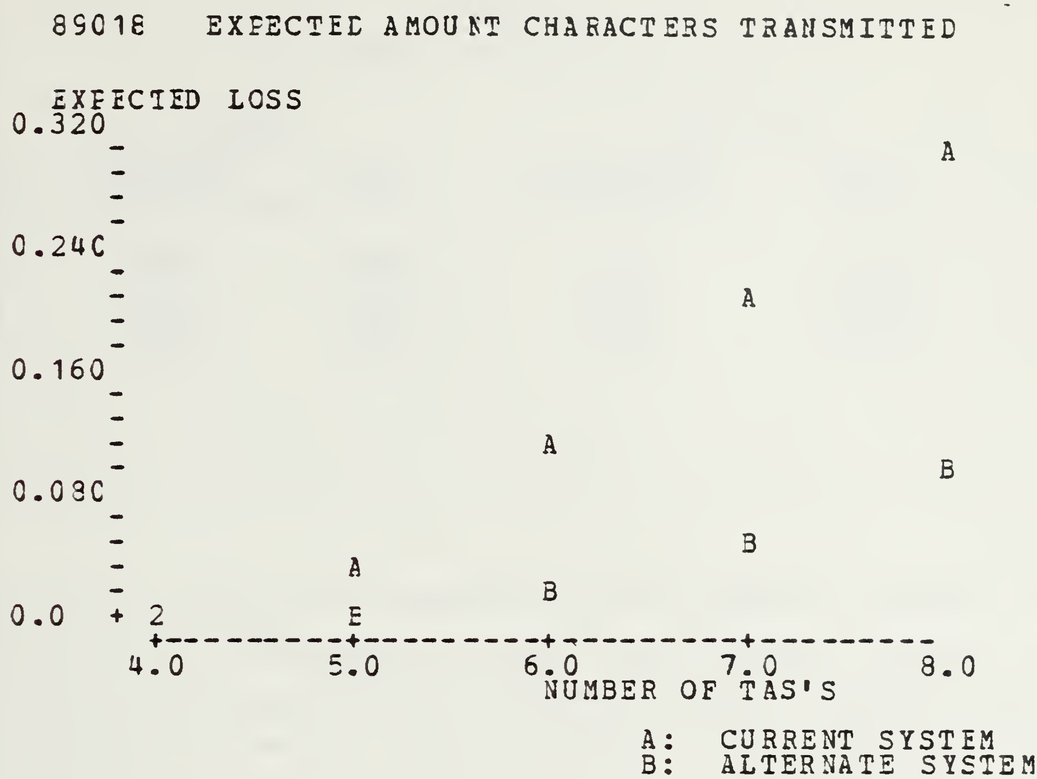


Figure 9.1 System times with increasing TASs.

This is saying while there is only 61% interactive work, the port allocation scheme involved will not be able to service 0.30 of the customers in an eight TAS environment. In the alternative method, the proportion of interactive use is 1.0 and the expected customer loss rises to 9%. The difference in the proportion of use between the two methods occurs



REGRESSION EQUATION FOR A = - 0.326 +0.0768 X1
S.D. ABOUT REGRESSION LINE IS 0.02175

REGRESSION EQUATION FOR B = - 0.103 +0.0218 X1
S.D. ABOUT REGRESSION LINE IS 0.01830

Figure 9.2 Expected loss rate.

because in method 1, both interactive and data transfer work occur on the 15 SOLIS ports while in method 2, only interactive processing occurs on the 14 SOLIS ports with all data transfers are handled by one high-speed port. The proportion of use remains at 0.61 for method 1 because the

TABLE VI

Proportion of interactive use

# TAS	current proportion of interactive use	current exptd loss	alternate proportion of interactive use	alternate exptd loss
4	0.60	0.00	1.00	0.00
5	0.61	0.04	1.00	0.00
6	0.61	0.12	1.00	0.01
7	0.61	0.22	1.00	0.04
8	0.61	0.30	1.00	0.09

division of work in a session is not being varied. The parameters for the amount of time the customer spends in the interactive portion and the amount of data being requested for transfer have not been changed. The variable being changed is the number of population sources to SOLIS. This is being increased from four to eight. The performance of method 2 is clearly preferable.

The amount of data transferred is the other parameter of interest. To study the impact of increasing batch demands on COINS, we started with a base number of TASs, and for each base number, iterated through the model while increasing the transfer demands. The base numbers used were five to eight. Just as we did in the original configuration of four TASs, the number of characters to be transferred is increased from 89,018 to 209900. Appendix C contains the summarized results of these runs. The relationships observed between the system time of the current method and system time1 and system time2 of the alternate approach generally holds. As the amount of data increases, system time1 increases from about 1 minute to 3 minutes over the

system time of method 1. And system time2 ranges between 4 to 9 minutes less than the system time of the current system. The interesting statistic is the changes in expected customer loss. For the current method, each time the amount of data transferred is incremented, there is a corresponding jump in the percentage of expected customer loss ranging from 1 to 3 percentage points. This phenomena is not observed in the alternate approach. The percentage of expected loss remains the same for all increases in transfer demands. These findings are condensed to one line entries for each network configuration in Table VII . For example, the second line of Table VII contains the condensed results for five TASs functioning under larger data transfers ranging from 89,018 to 209,900 characters. The loss column indicates a range of 4% to 21% expected customer

TABLE VII

Ranges of system times and expected loss

	CURRENT		ALTERNATE		
# TAS	SYSTEM TIME	EXPECTED LOSS	SYSTEM TIME1	SYSTEM TIME2	EXPECTED LOSS
4	12.43-19.45	.00-.00	13.17-21.70	8.01- 9.52	.00-.00
5	12.33-19.29	.04-.21	13.24-22.39	8.11-10.30	.00-.00
6	12.24-19.23	.12-.34	13.23-22.29	8.13-10.29	.01-.01
7	12.30-19.16	.22-.44	13.29-22.32	8.19-10.30	.04-.04
8	12.30-19.39	.30-.53	13.35-22.30	8.22-10.21	.09-.09

times are in minutes

loss. Table VIII is the companion to Table VII The table shows the range of proportion of interactive use together with expected customer loss. For the current method,

TABLE VIII

Ranges of proportion of interactive use

# TAS	current proportion of interactive use	expected loss	alternate proportion of interactive use	expected loss
4	0.62-0.51	0.00-0.00	1.00	0.00-0.00
5	0.61-0.50	0.04-0.21	1.00	0.00-0.00
6	0.61-0.50	0.12-0.34	1.00	0.01-0.01
7	0.61-0.51	0.22-0.44	1.00	0.04-0.04
8	0.61-0.50	0.30-0.53	1.00	0.09-0.09

proportion of use is about 0.61 when the expected number of characters to transfer is 89,018. As the number of characters is increased to 209,000, the proportion of use drops to about 0.50. In method 1, it is quite apparent that as more of the session time is used for data transfers, the price is increased expected customer loss.

Appendix D has ten sets of graphs for the ten different data transmission amounts. For each set, there is one graph showing the relationships between the system times and another graph comparing expected loss as the number of TASs is increased. For each expected loss graph, the calculated regression line and error about the calculated line is shown. Although customer loss also occurs in method 2, the slope of the expected loss line for method 2 is consistently smaller than the slope of method 1, as seen in the plots in Appendix D.

Close examination of the numbers describing the performance of method 2, reveals that expected customer loss is not necessarily because of larger data transfers, but rather the configuration is approaching its limits in satisfying

the purely interactive demands. This is consistent with our findings in Table VII, where given a network environment and a certain customer interactive work-profile, no fluctuations are observed in expected loss for the alternate method, as increases in the data transfer sizes are made. Referring to the model of SOLIS as a 2-node network in tandem, provides an explanation of this phenomena. Recall that method 2 is simply a reallocation of network capacity of method 1, whereby 14 of the available 15 SOLIS ports are devoted only to interactive work and all batch work is conducted on one port. Increased data transfer sizes only impacts node 2.

It was of interest to uncover the kind of situation that would result in expected customer loss. A closer look was taken when TAS5 is introduced into COINS. The approach taken was for each increase in data size transfer, the arrival rate was varied from 2.68 to 17.68 customers per hour. Appendix E contains the results of these runs. They indicate that a configuration of four operational TASs with a fifth functioning at a rate of 5.10 customers per hour, and expected data transfer size of 107,712 characters, the network can expect to lose 1% of the customers.

We feel the analysis certainly indicates that a reallocation of the interactive ports performs better than the present method. Nevertheless, there are other concerns the COINS/PMC must address before deciding on this reallocation scheme. These lie in the area of implementation. We have provided the conceptual basis that reallocation is better and have not looked at the price of implementation. Although examining implementation costs is beyond the scope of this thesis, we are compelled to mention the more important aspects contributing to this cost. In our view, host to host protocol development and network integration are the most serious issues. Converting SOLIS from a 2-stage facility to a 2-node network in tandem requires processing

intelligence in both SOLIS and the TASs. This needs to be carefully specified on the host as well as on the application level.

After the issue of protocols has been addressed, the problems of network integration becomes paramount. A well thought out transition plan must be developed, whereby network perturbation is kept at an acceptable level. A transition generally suggests operating in the old and new systems in parallel. The COINS/PMO has experience in this area, since the network transitioned from a star store-and-forward switch network to one of packet-switching. Dual services were maintained until all nodes were ready to operate at the new level.

X. CONCLUSIONS AND RECOMMENDATIONS

A. SUMMARY

1. Current Environment

Within an operational network environment of four TASs and one interactive database host (SOLIS), the alternate proposal has a work completion time 4.5 minutes less than with the current method. This is an advantage if concern is in freeing an interactive path for a new interactive search and refinement session or getting the data back to the TAS for follow-on processing of information fusion or editing. The alternate proposal takes about a minute longer than the present method to have the data printed at the terminal site. From a proportion of use standpoint, the portion of the interactive capactiy of method 2 when used, is completely devoted only to interactive work. However, in method 1, only 0.61 of the interactive capacity is utilized for interactive work. The remaining 0.39 is used for file transfer functions. Despite this fact, the work demands of four TASs with SOLIS do not indicate any expected customer loss in the present configuration and certainly not in the alternate proposal.

2. Population Growth

Customer growth was considered as more TASs were added. A transform of SOLIS from a 2-stage service facility to a 2-node network in tandem exhibits robustness as the workload increases. It is less sensitive to growth than the current method. As the number of TASs was increased from four to eight, the completion time to a TAS was about 4 minutes less in method 2 than in method 1, while completion

time to a customer terminal was about a minute longer in the alternate than in the current method. In terms of expected customer loss and proportion of use, method 2 displayed an expected loss range of zero at four TASs to 0.09 at eight TASs with a proportion of use of 1.0. The current approach exhibited an expected loss range of zero at four TASs to 0.30 at eight TASs with a proportion of use of 0.61.

3. Data Transfer Growth

Examining growth in terms of larger file transfers, the current method was considerably more sensitive to changes in filesize than the alternative approach. When the file transfer size is increased from 89,018 to 209,900 characters, completion time to a TAS in method 2 is consistently 4 to 9 minutes shorter than method 1. Completion time to a terminal in method 2 ranged from 1 to 3 minutes longer than the current method. Similarly, while the proportion of use remained at 1.0 for the alternate method, this value dropped from 0.61 to 0.50 for the current method for all environments as the transfer size increased. An expected customer loss of 0.04 is first noticed in the five TAS environment for method 1 as transfer size was increased, while an expected loss of 0.01 is first observed in a six TAS configuration for method 2. For each network configuration from five to eight TASs as the data size is increased, the expected customer loss in method 1 increases while the loss remains constant for method 2. For example, in a six TAS environment, as data transfer size is increased, the expected loss in method 1 ranges from 0.12 to 0.34 while expected loss of method 2 remains constant at 0.01.

4. Review

Table IX was prepared to provide a summarized view of proportion of use and expected loss when the number of TASs and data transfer amounts are varied. In the current method, the proportion of use metric is sensitive only to data transfer increases. The alternate approach shows no variation in this variable. Since expected loss is observed in both methods as each of the parameters is varied, we can only ask the question how much better is one from the other. It is quite evident that method 2 is substantially more

TABLE IX

Condensed Comparison Chart

	PROPORTION OF INTERACTIVE USE		LOSS	
	CURRENT	ALTERNATE	CURRENT	ALTERNATE
TAS growth (4 to 8)	.61-.61	1.0-1.0	0-30%	0-9%
Data growth (89,018 to 209,900)	.61-.50	1.0-1.0	0-53%	0-9%

stable than the current method.

B. CONCLUSIONS

Job specialization has frequently been the path to higher efficiency and better performance as systems grow. So, it is no great surprise that as customer population

growth and demand for larger data transfers are realized, a configuration whose underlying philosophy is one of specialization would perform better than a non-specialized system. Customer work-profile is a key parameter in the evaluation just performed. The benefits and advantages of this reallocation scheme can only be realized if the customer work-profile remains approximately the same as the empirical data suggests or if the profile changes in favor of larger data transfers. In other words, if the profile changes to where users are spending more time in the interactive mode than in the batch mode, the new arrangement may not improve services and in extreme cases will decrease performance.

C. RECOMMENDATIONS

In light of customer population growth alone, some consideration should be given to reallocation. However, when both population and data transfer growth are anticipated, we recommend serious consideration of this new allocation scheme. This involves efforts in the development and evaluation of host and process level protocols and a carefully designed implementation plan addressing the problems of operating in a period of transition.

APPENDIX A

EMPIRICAL DATA ANALYSIS

TABLE X

SOLIS interactive-only time analysis

solis interactive time analysis
(no hard-copy request)

chi-squared goodness of fit test (exponential)

n= 68 mean = 10.13 minutes

mins.	#obs	p	E(x)	chi-squared statistic
0- 5	28	.39	26.5	.08
6-10	13	.24	16.3	.67
11-15	13	.14	9.5	1.29
16-20	7	.09	6.1	.13
21-25	3	.05	3.4	.05
26-30	2	.03	2.0	0.00
31-35	2	.02	1.4	.26

df = 5 chi-square = 2.47
alpha critical > .25

TABLE XI

SOLIS interactive time analysis

solis interactive time analysis
(with hard-copy request)

chi-squared goodness of fit test (exponential)

n=212 mean = 6.2 minutes

mins.	#obs	p	E(x)	chi-squared statistic
0- 5	133	.62	131.17	.02
6-10	43	.21	44.63	.05
11-15	19	.09	19.98	.04
16-20	7	.04	8.94	.42
21-25	4	.02	4.00	0.0
26-30	4	.01	2.12	1.66
31-35	2	.003	.80	1.80

df = 5 chi-square = 3.99
alpha critical > .25

TABLE XII

SOLIS data transfer time analysis

chi-squared goodness of fit test (exponential)

n=191 mean = 6.8 minutes

mins.	#obs	p	E(x)	chi-squared statistic
1- 5	109	.52	99.3	.947
6-10	45	.25	47.7	.152
11-15	20	.12	22.9	.367
16-20	9	.06	11.9	.540
21-25	5	.03	5.7	.085
26-30	1	.01	1.9	.426
31-35	2	.01	1.9	.005

df = 5 chi-square = 2.52
alpha critical > .25

TABLE XIII

TAS1 inter-arrival time analysis

chi-squared goodness of fit test (exponential)

n=174 mean = 3.393 minutes

mins.	#obs	p	E(x)	chi-squared statistic
0-2	95	.579	100.7	.32
3-5	45	.24	41.76	.25
6-8	18	.10	17.40	.02
9-11	8	.04	6.96	.15
12-14	6	.02	3.48	1.82
15-17	1	.01	1.74	.31
18-20	1	.003	.52	.44

df = 5 chi-square = 3.31
alpha critical > .25

TABLE XIV

TAS2 inter-arrival time analysis

chi-squared goodness of fit test (exponential)

n=118 mean = 6.52 minutes

mins.	#obs	p	E(x)	chi-squared statistic
0- 5	58	.53	63.1	.42
6-10	36	.24	29.35	1.5
11-15	9	.11	13.64	1.5
16-20	7	.05	6.34	.06
21-25	3	.02	2.9	0.0
26-30	2	.01	1.36	.29
31-35	2	.005	.63	2.9
>36	1	.004	.55	.36

df = 6 chi-square = 7.03
alpha critical > .25

TABLE XV

TAS3 inter-arrival time analysis

chi-squared goodness of fit test (exponential)

n= 81 mean = 11.76 minutes

mins.	#obs	p	E(x)	error
0- 7	41	.49	39.9	.026
8-15	20	.249	20.2	0.0
16-23	8	.126	10.25	.49
24-31	6	.064	5.19	.124
32-39	3	.032	2.63	.05
40-47	2	.016	1.33	.33
>48	1	.014	1.17	.026

df = 5 chi-square = 1.046
alpha critical > .25

TABLE XVI

TAS4 inter-arrival time analysis

chi-squared goodness of fit test (exponential)

n= 29 mean = 22.82 minutes

mins.	#obs	p	E(x)	error
0-21	15	.618	17.9	.469
22-43	10	.235	6.8	1.50
44-65	2	.089	2.6	.13
66-87	2	.034	.98	1.06

df = 2 chi-square = 3.16
alpha critical > .05

APPENDIX B

OPERATING INSTRUCTIONS FOR INS MODEL

There are 3 categories of parameter input for the INS model. Information is required describing host(s) characteristics, the network environment and finally TAS and server-TAS descriptions.

A. HOST CHARACTERISTICS

For each interactive host, the following information must be input:

1. proportion of customers doing only interactive work. That is, those customers that will not request a hard-copy output.

   FORMAT: real. For example: .24

2. distribution describing the interactive-only session time and the parameters for that distribution. This is a series of 3 fields.

   field 1: must be integer value of 1 or 2 or 3, depending on the distribution.

   1 = uniform

   2 = normal

   3 = exponential

   FORMAT: integer. For example 3

   field 2 and 3: parameters for the distribution and they must be in minutes.

   if uniform: n1 and n2, where distribution is uniform between n1 and n2.

   FORMAT: real. For example 5.2 10.3

if normal: n1 and n2, where n1 is the mean and n2 is the standard deviation.

FORMAT: real. For example 6.8 2.5

if exponential: n1, where n1 is the mean. Note for this case, some number must be input for n2 even though that is meaningless. This is because of a minor inflexibility in the program structure.

FORMAT: real. For example 6.2

3. distribution describing the amount of characters requested in a hard-copy command, and the parameters for that distribution. This too is a series of 3 fields. Their formats are as those described above.

4. distribution describing the interactive time when a hard-copy request is made, and the parameters for that distribution. The parameters must be in minutes. This is a series of 3 fields and their formats are exactly as those described above.

5. hi-speed flag: used to indicate whether or not the host is to be considered as having a hi-speed transfer facility when the alternate configuration is run.

0 = no hi-speed facility

1 = yes

6. number of ports: this is the number of interactive ports the host is offering to the network.

FORMAT: integer. For example 15

B. NETWORK CONFIGURATION

This is the number of TASs, HOSTs and server-TASs in the network. The numbers must be input in that order.

FORMAT: integer. For example 4 1 0

Comment: if server-TASs are not to be considered as offering network services, then the number of server-TASs must be zero. However, if it still functions as a TAS, they should be included in the number of TASs.

C. TAS AND SERVER-TAS CHARACTERISTICS

The fields and formats describing the TAS and the server-TAS are exactly the same. TAS characteristics must be input first, followed by the server-TAS. This is so because the program first builds the data structures for the TASs, then the server-TASs. The unique thing about the server-TASs is that they also function as HOSTs, hence they are also duplicated in the HOST data structures with an extra flag-field indicating that they are really a server-TAS. This flag-field is for port accounting purposes. If a customer from 1 server-TAS requests access to a network resource, its accounting tables as a TAS are updated to reflect this. However, its accounting tables as a HOST must also be updated to reflect a busy condition. The following information must be input:

1. customer arrival rate per hour.

   FORMAT: real. For example, 17.68

2. number of ports available for interactive work.

   FORMAT: integer. For example, 24

3. proportion of users requesting network access.

   FORMAT: real. For example .70

4. The next 3 fields describe the proportion of customers wanting access to SOLIS, USIS, and NUIS. Because of a minor shortcoming of the way the program was written, these numbers must be cumulative. For example, if there are .80 going to SOLIS, .10 going to USIS and .10 going to NUIS, this information must be input as

   0.80 .90 1.0

   FORMAT: real. For example .80 .90 1.0

5. hi-speed flag: used to indicate whether or not the TAS or server-TAS is to be considered as having a hi-speed transfer facility when the alternate configuration is run.

0 = no hi-speed facility

1 = yes

APPENDIX C
TABLES OF RESULTS

The following four Tables show performance measurements of a network configuration with five, six, seven and eight TAS's as the data transfer amount is increased.

TABLE XVII
Five TAS configuration

	current				alternate	
# char.	system time	exptd loss	pro.of use	system time1	system time2	exptd loss
89018	12.33	.040	.610	13.24	8.11	.020
97920	12.84	.050	.600	13.88	8.24	.020
107712	13.41	.060	.590	14.60	8.39	.020
118483	14.04	.070	.570	15.39	8.56	.020
130331	14.71	.090	.560	16.27	8.76	.020
143364	15.46	.110	.550	17.24	8.98	.020
157701	16.32	.130	.540	18.33	9.24	.020
173471	17.22	.150	.530	19.52	9.53	.020
190818	18.18	.300	.520	20.87	9.88	.020
209900	19.29	.340	.510	22.39	10.30	.020

times are in minutes

TABLE XVIII

Six TAS configuration

# char.	current system time	current exptd loss	current pro.of use	alternate system time1	alternate system time2	alternate exptd loss
89018	12.24	.120	.610	13.23	8.13	.011
97920	12.75	.130	.600	13.87	8.27	.011
107712	13.29	.150	.590	14.58	8.42	.011
118483	13.90	.170	.570	15.37	8.59	.011
130331	14.59	.200	.560	16.24	8.79	.011
143364	15.31	.220	.550	17.22	9.03	.011
157701	16.16	.250	.540	18.30	9.28	.011
173471	17.06	.280	.530	19.50	9.58	.011
190818	18.00	.300	.520	20.82	9.91	.011
209900	19.23	.340	.500	22.29	10.29	.011

times are in minutes

TABLE XIX

Seven TAS configuration

# char.	current system time	current exptd loss	current pro.of use	alternate system time1	alternate system time2	alternate exptd loss
89018	12.30	.22	.61	13.29	8.19	.04
97920	12.79	.24	.60	13.97	8.33	.04
107712	13.33	.26	.59	14.66	8.49	.04
118483	13.98	.28	.57	15.45	8.66	.04
130331	14.66	.31	.56	16.33	8.86	.04
143364	15.43	.33	.55	17.29	9.08	.04
157701	16.21	.36	.54	18.37	9.33	.04
173471	17.13	.39	.53	19.55	9.61	.04
190818	18.23	.42	.51	20.87	9.94	.04
209900	19.16	.44	.51	22.32	10.30	.04

times are in minutes

TABLE XX

Eight TAS configuration

# char.	current system time	current exptd lcss	pro.of use	alternate system time1	alternate system time2	alternate exptd lcss
89018	12.30	.30	.61	13.35	8.22	.09
97980	12.85	.33	.60	14.01	8.37	.09
107712	13.38	.35	.58	14.72	8.52	.09
118483	13.98	.37	.57	15.51	8.69	.09
130331	14.69	.40	.56	16.38	8.88	.09
143364	15.47	.42	.55	17.34	9.09	.09
157701	16.25	.45	.54	18.41	9.33	.09
173471	17.14	.47	.53	19.58	9.60	.09
190818	18.22	.50	.52	20.86	9.88	.09
209900	19.39	.53	.50	22.30	10.21	.09

times are in minutes

APPENDIX D

SYSTEM TIME AND EXPECTED LOSS CHARTS

There are a pair of graphs for each increment of data transfer sizes. The first graph shows the relationships between the system times of the two methods. The second is a plot of the expected loss of the two approaches. For both graphs, the x-axis is the number of TAS's operating in the network with arrival rate of 17.68 customers per hour. All times are in minutes.

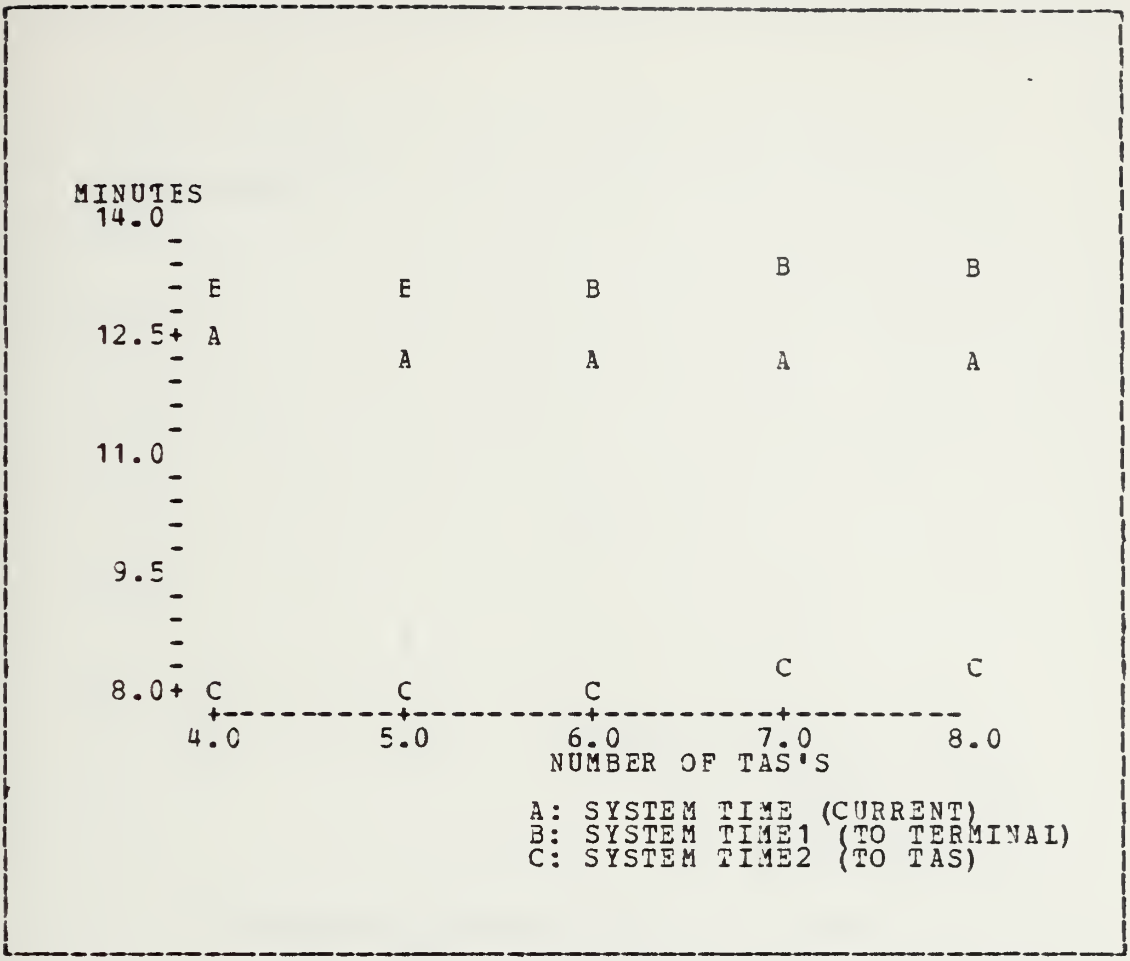
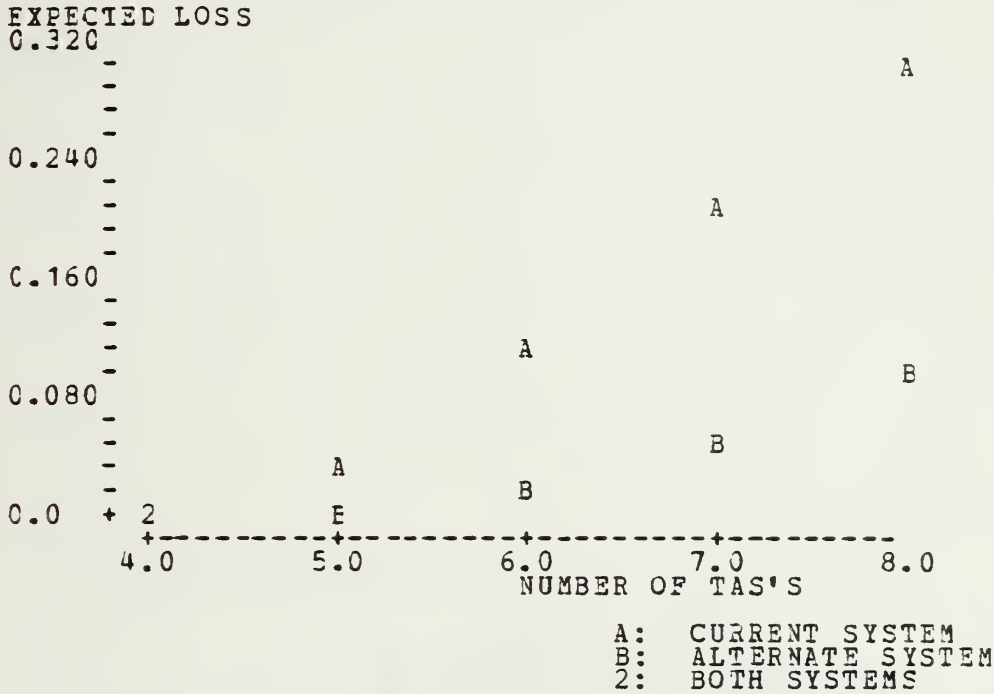


Figure D.1 System times with 89,018 characters.



REGRESSION EQUATION FOR A = - 0.326 +0.0768 X1
S.D. ABOUT REGRESSION LINE IS S = 0.02175

REGRESSION EQUATION FOR B = - 0.103 +0.0218 X1
S.D. ABOUT REGRESSION LINE IS S = 0.01830

Figure D.2 Expected loss with 89,018 characters.

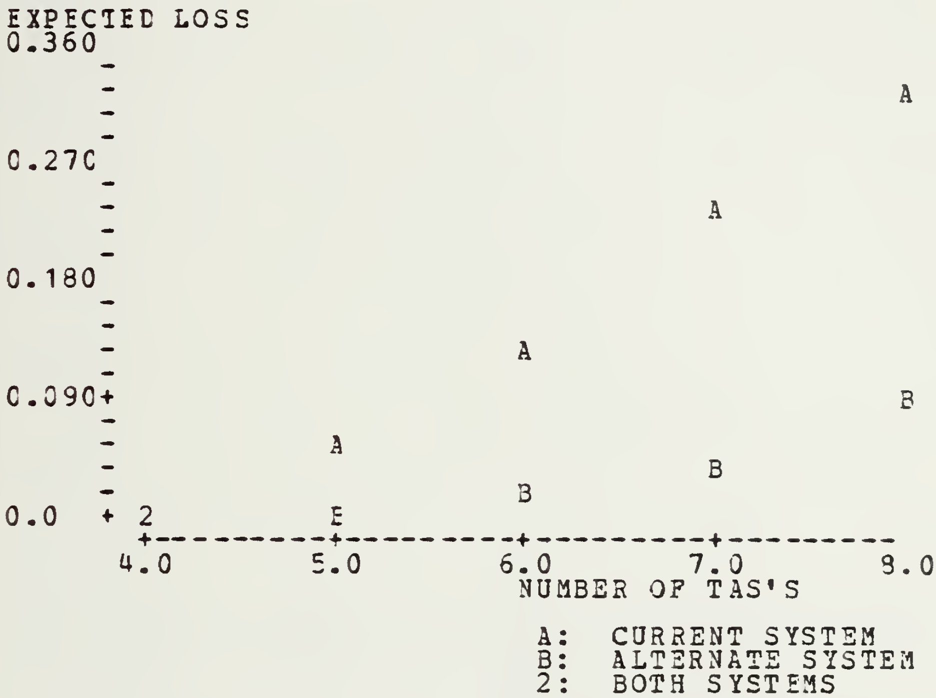
```
      97920    EXPECTED AMOUNT CHARACTERS TRANSMITTED

    MINUTES
     14.0+                E             B             B             B
         -  B
         -
         -
         -  A             A             A             A             A
     12.5
         -
         -
         -
         -
     11.0
         -
         -
         -
         -
      9.5
         -
         -
         -
         -  C             C             C             C             C
      8.0
            +-------------+-------------+-------------+----------
           4.0           5.0           6.0           7.0           8.0
                                      NUMBER OF TAS'S

                                A:  SYSTEM TIME  (CURRENT)
                                B:  SYSTEM TIME1  (TO TERMINAL)
                                C:  SYSTEM TIME2  (TO TAS)
```

Figure D.3 System times with 97,920 characters.

97920 EXPECTED AMOUNT CHARACTERS TRANSMITTED



REGRESSION EQUATION FOR A = - 0.350 +0.0832 X1
S.D. ABOUT REGRESSION LINE IS 0.02071

REGRESSION EQUATION FOR B = - 0.103 +0.0218 X1
S.D. ABOUT REGRESSION LINE IS 0.01830

Figure D.4 Expected loss with 97,920 characters.

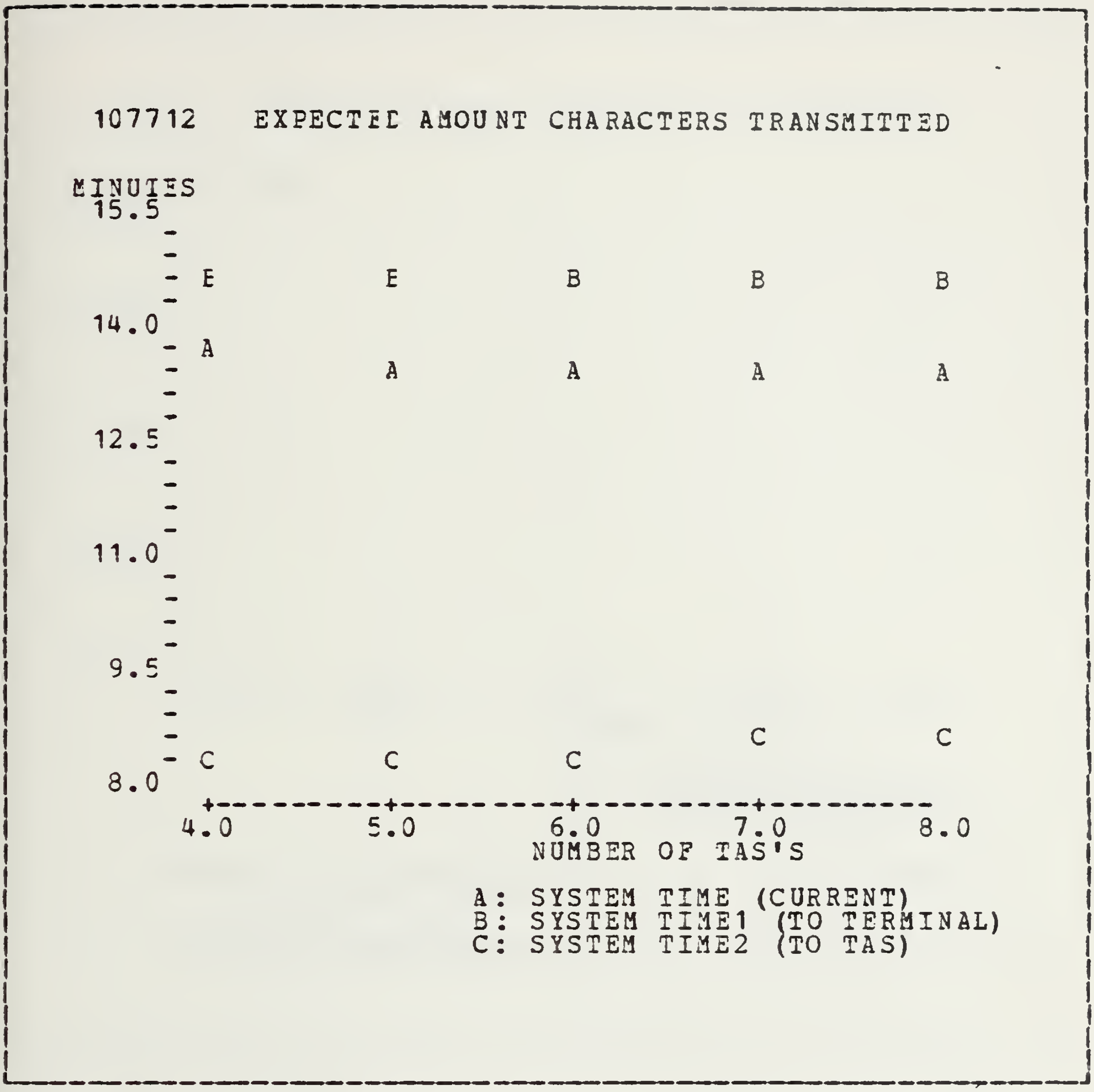


Figure D.5 System times with 107,712 characters.

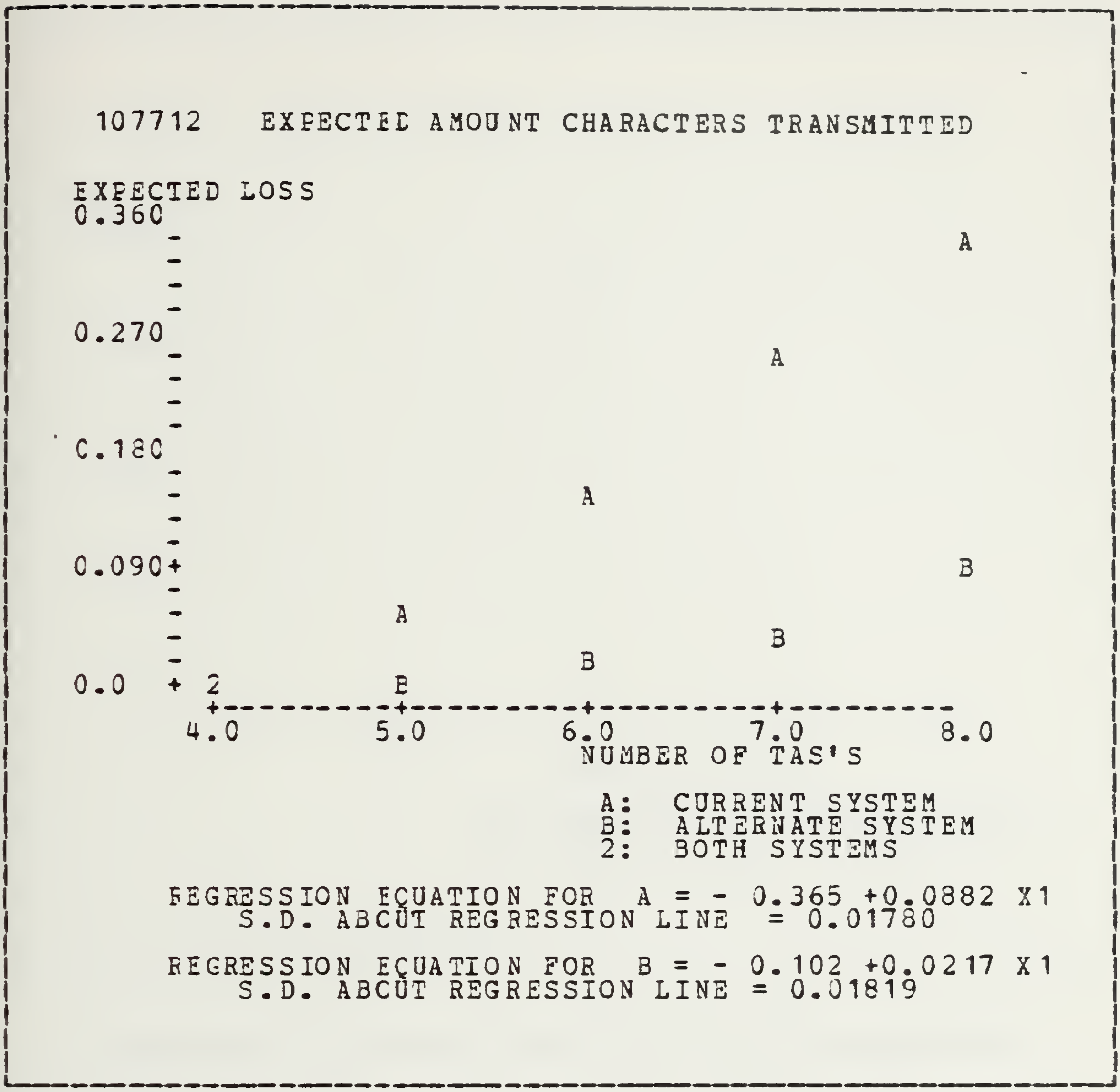


Figure D.6 Expected loss with 107,712 characters.

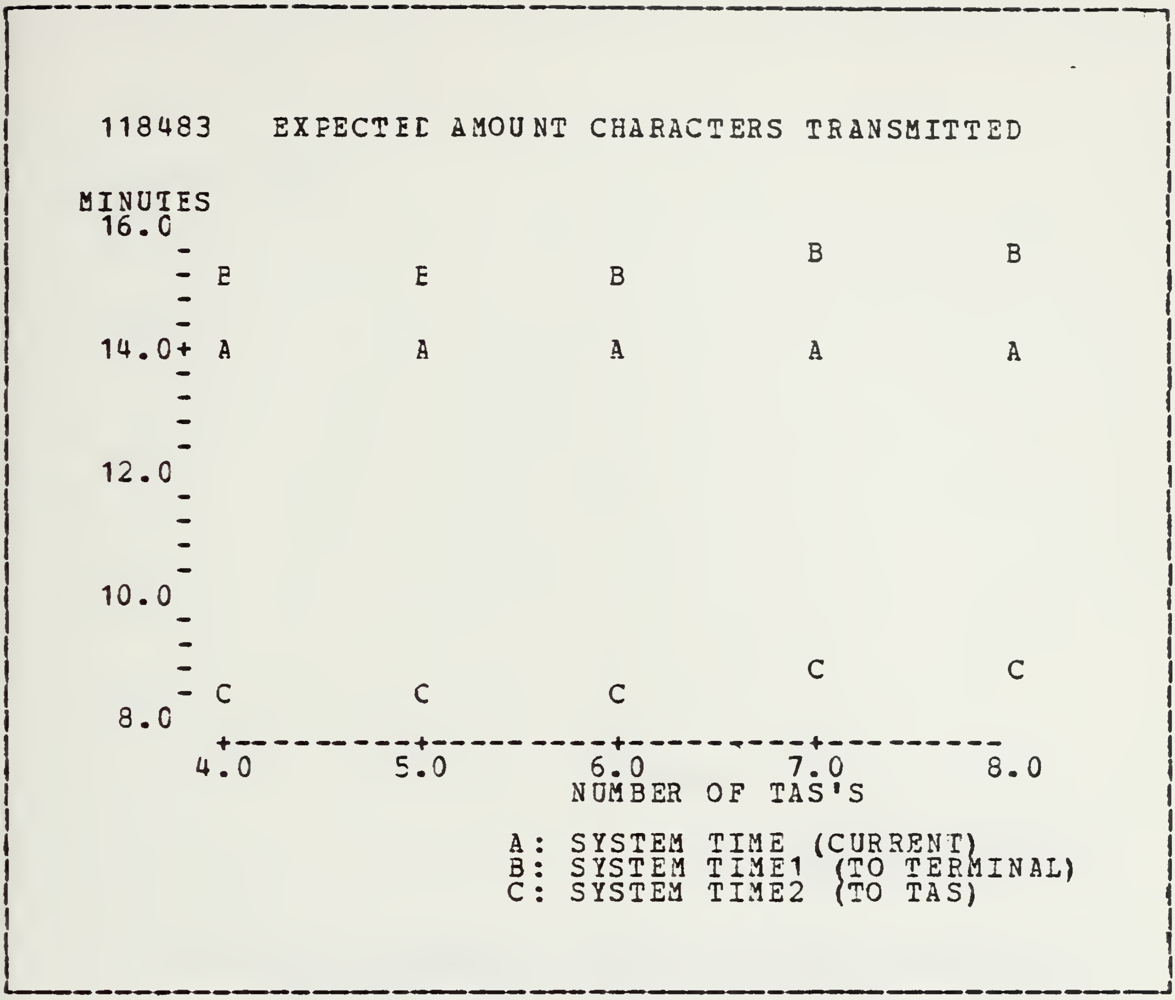
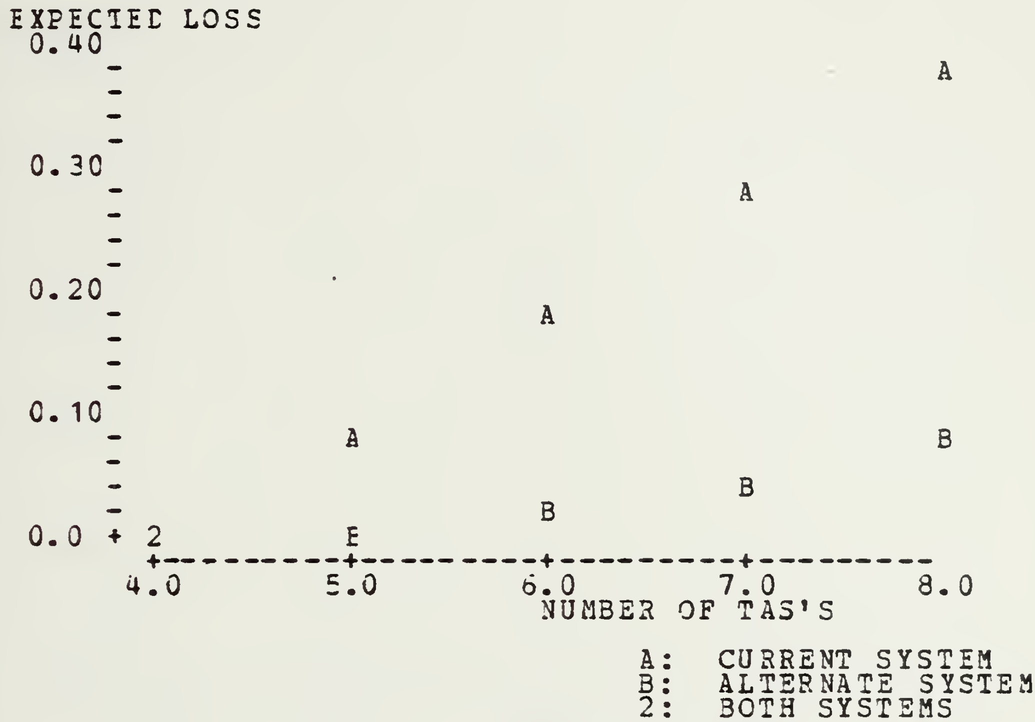


Figure D.7 System times with 118,483 characters.



REGRESSION EQUATION FOR A = - 0.378 +0.0930 X1
S.D. ABOUT THE REGRESSION LINE = 0.01409

REGRESSION EQUATION FOR B = - 0.102 +0.0217 X1
S.D. ABOUT THE REGRESSION LINE = 0.01819

Figure D.8 Expected loss with 118,483 characters.

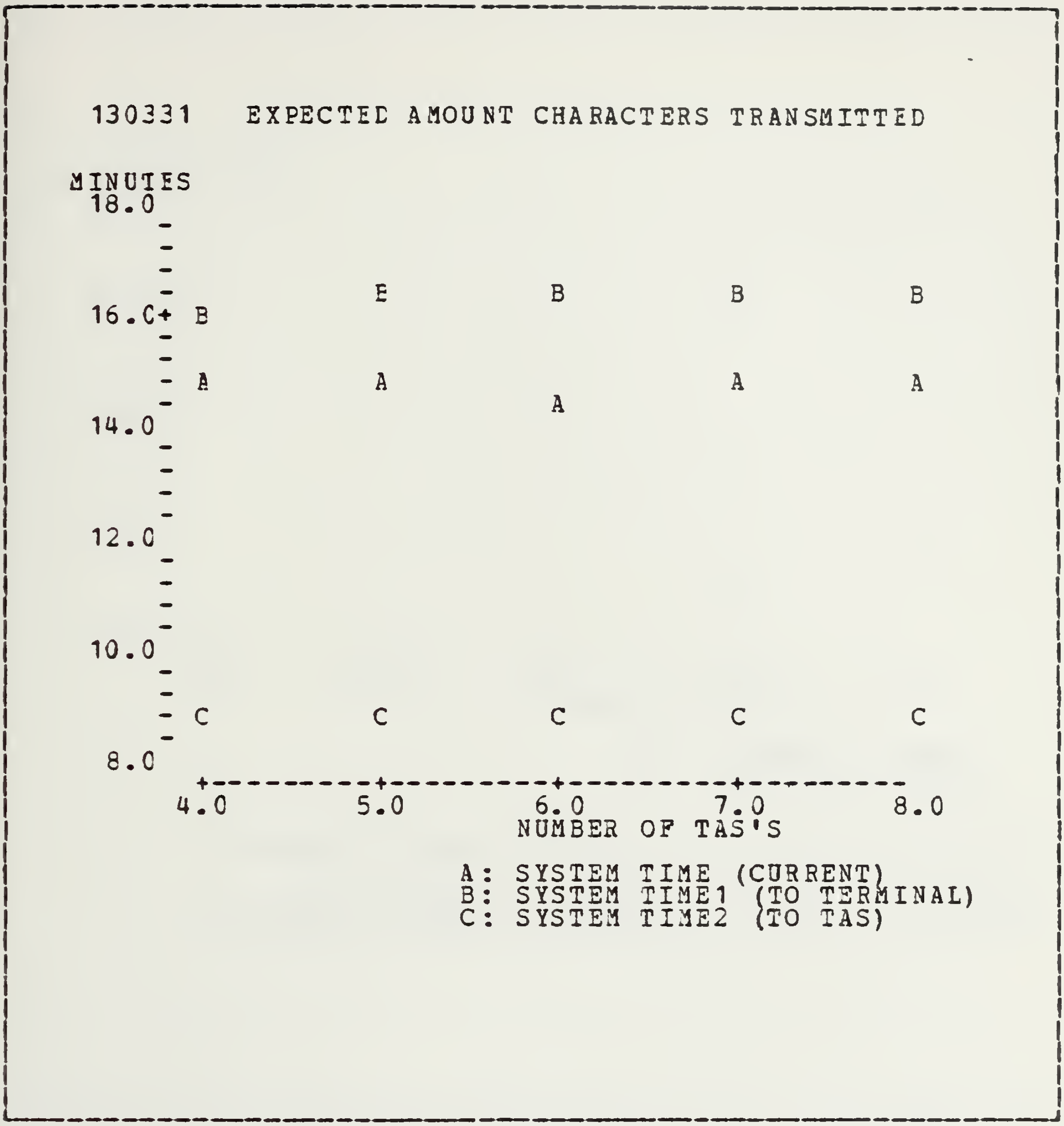
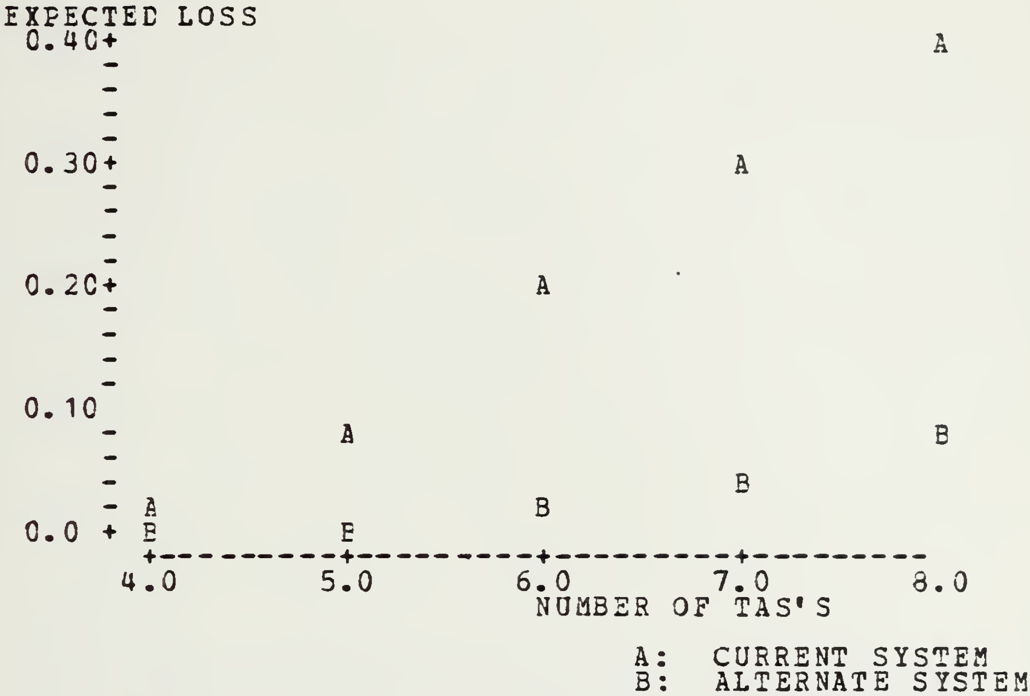


Figure D.9 System times with 130,331 characters.



REGRESSICN EQUATION FOR A = - 0.393 +0.0988 X1
S.D. ABOUT THE REGRESSION LINE = 0.01087

REGRESSICN EQUATION FOR B = - 0.102 +0.0217 X1
S.D. ABOUT THE REGRESSION LINE = 0.01787

Figure D.10 Expected loss with 130,331 characters.

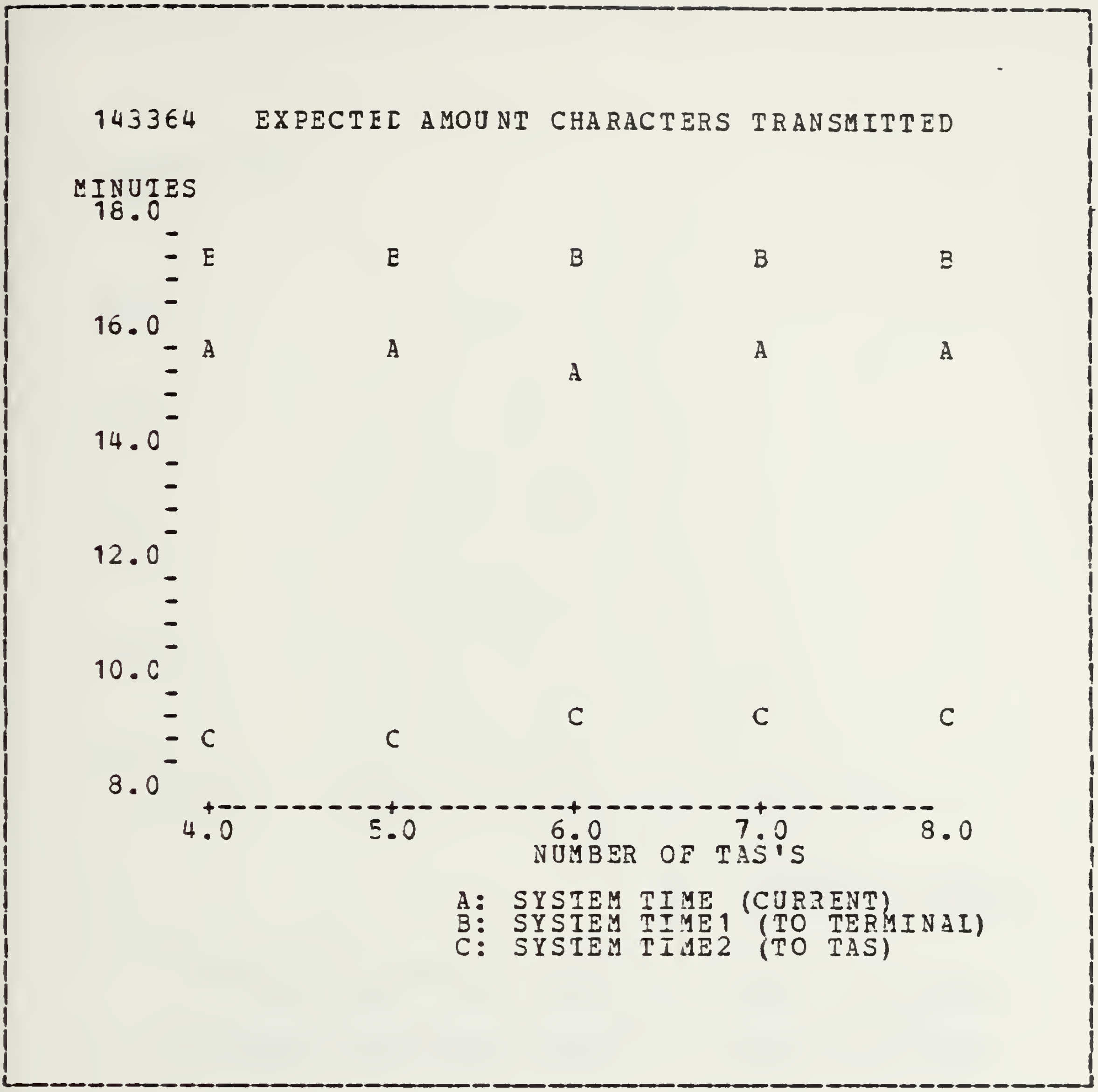
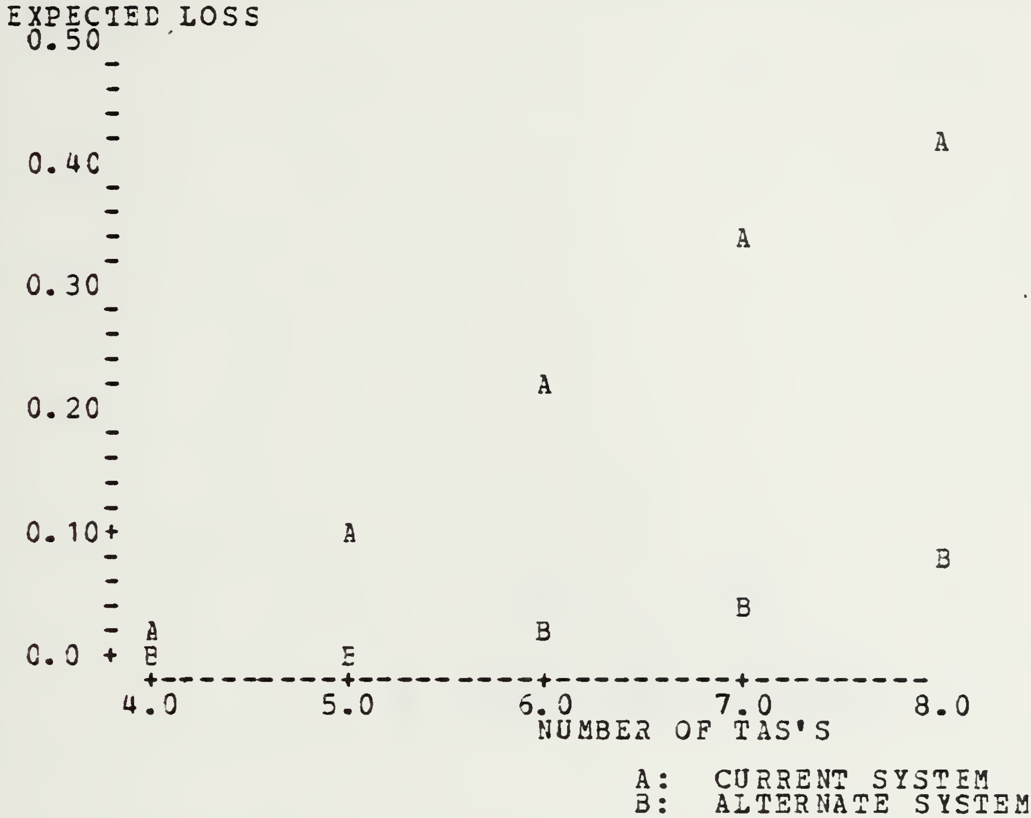


Figure D.11 System times with 143,364 characters.



REGRESSION EQUATION FOR A = - 0.393 +0.103 X1
S.D. ABOUT THE REGRESSION LINE = 0.008702

REGRESSION EQUATION FOR B = - 0.102 +0.0217X1
S.D. ABOUT THE REGRESSION LINE = 0.01819

Figure D.12 Expected loss with 143,364 characters.

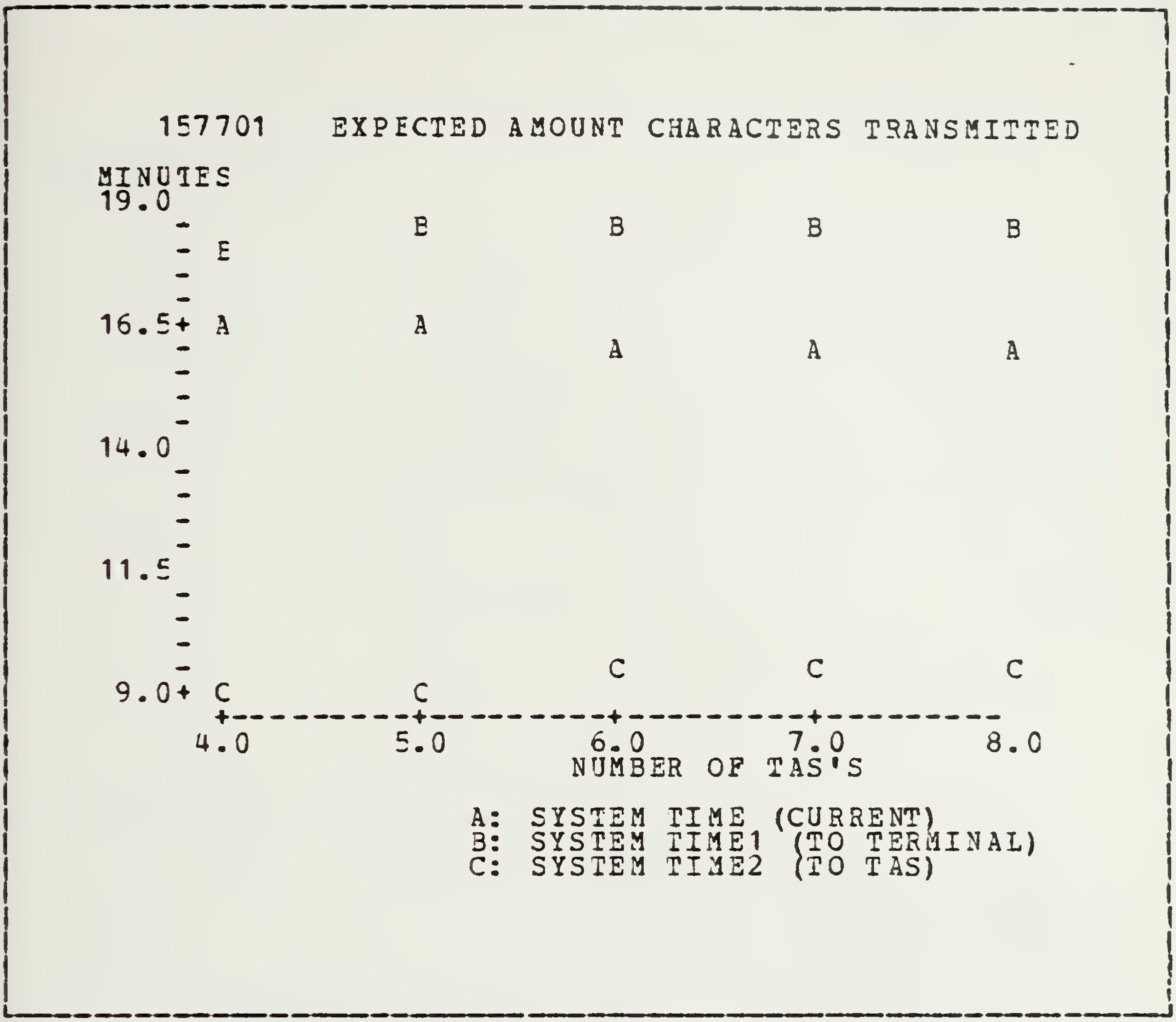
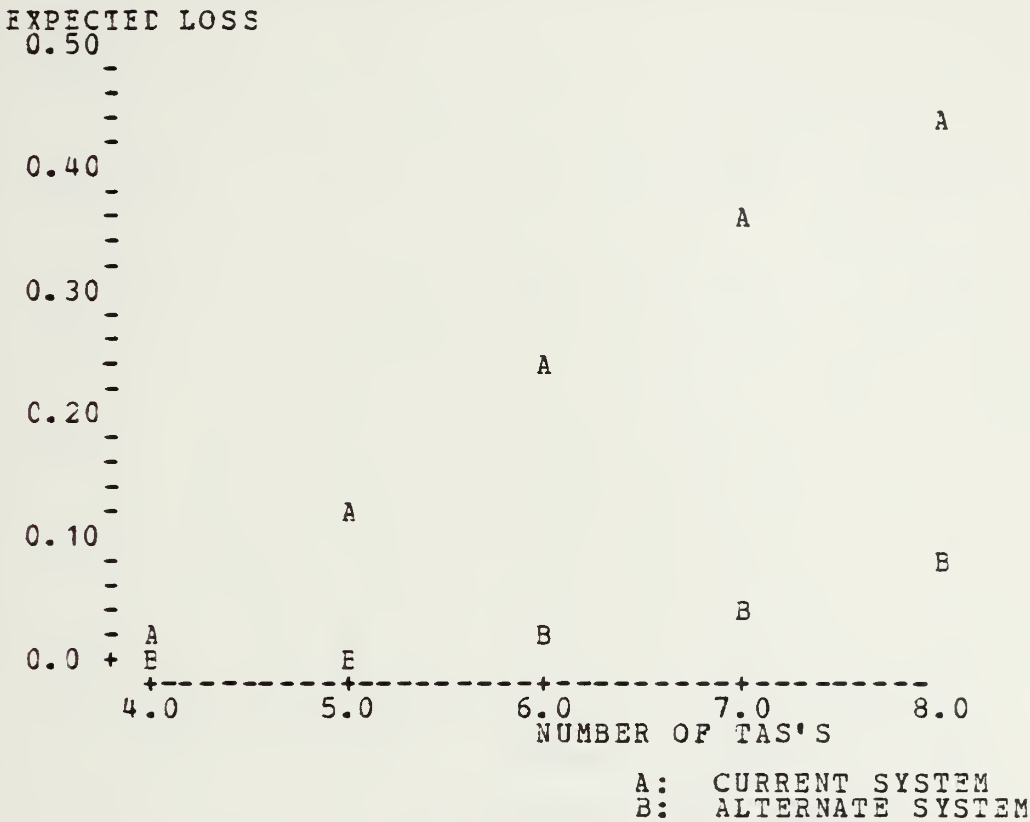


Figure D.13 System times with 157,701 characters.



REGRESSION EQUATION FOR A = - 0.403 +0.107 X1
S.D. ABOUT THE REGRESSION LINE = 0.0099

REGRESSION EQUATION FOR B = - 0.102 +0.0217X1
S.D. ABOUT THE REGRESSION LINE = 0.0181

Figure D.14 Expected loss with 157,701 characters.

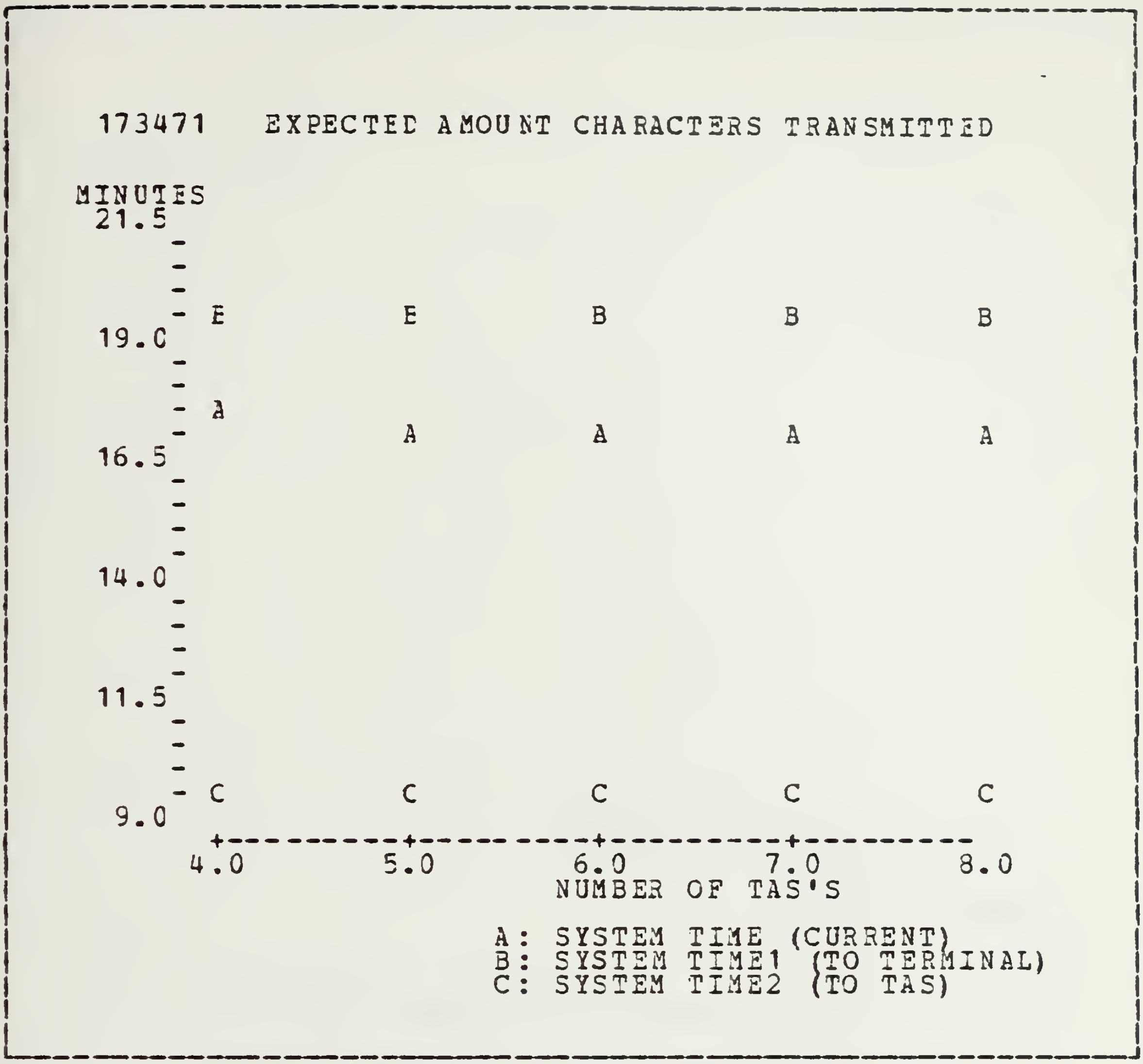
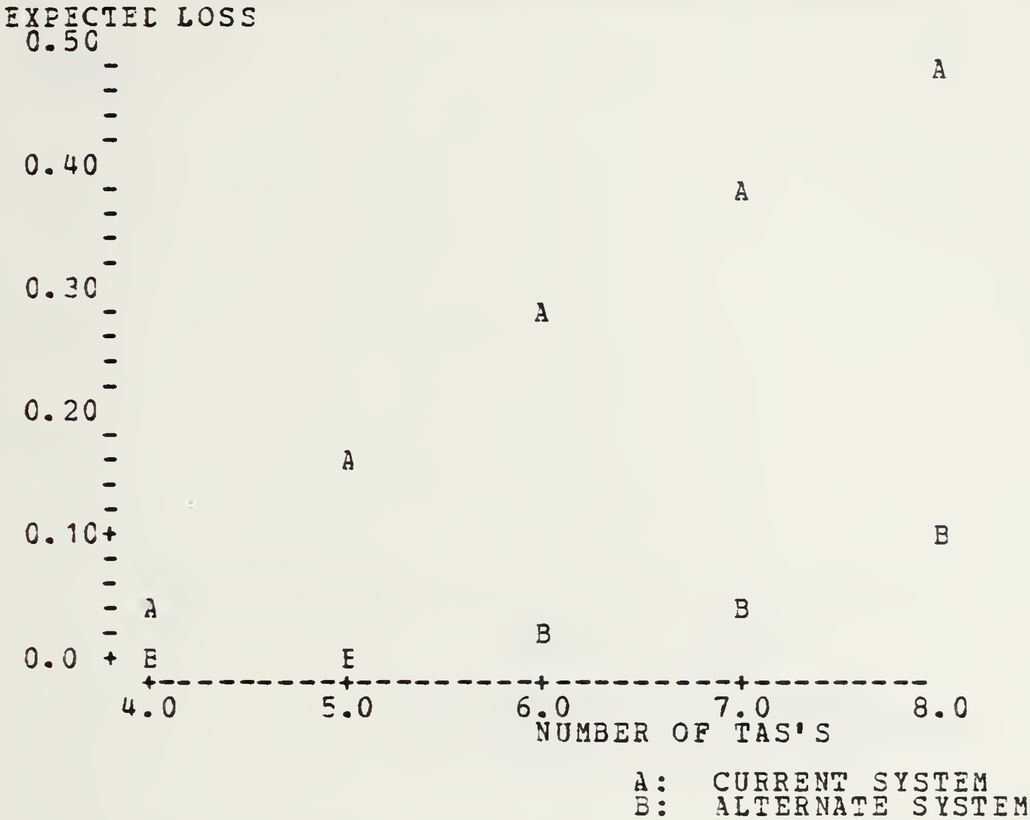


Figure D.15 System times with 173,471 characters.



REGRESSION EQUATION FOR A = - 0.398 +0.110 X1
S.D. ABOUT THE REGRESSION LINE = 0.0148

REGRESSION EQUATION FOR B = - 0.103 +0.021 X1
S.D. ABOUT THE REGRESSION LINE = 0.0185

Figure D.16 Expected loss with 173,471 characters.

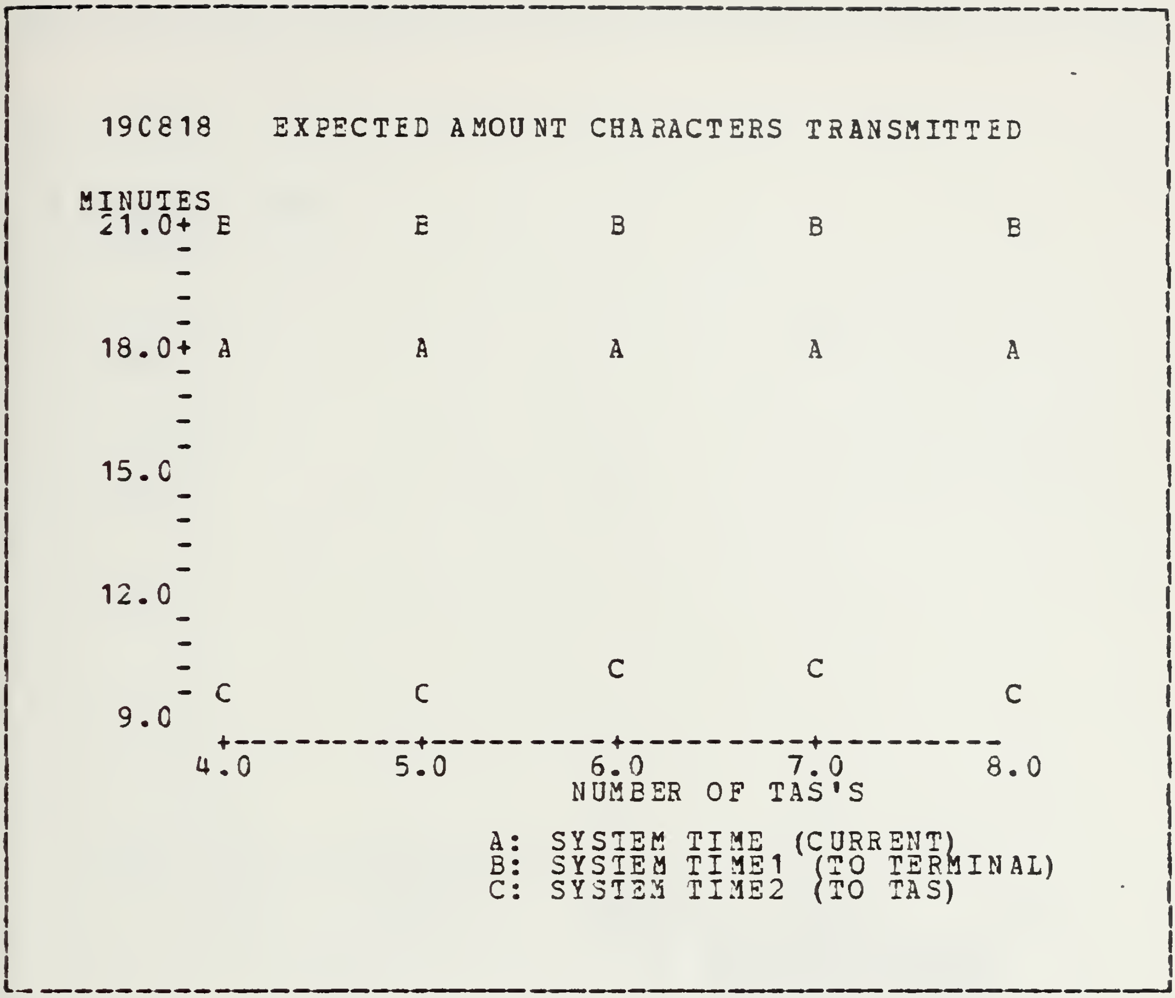
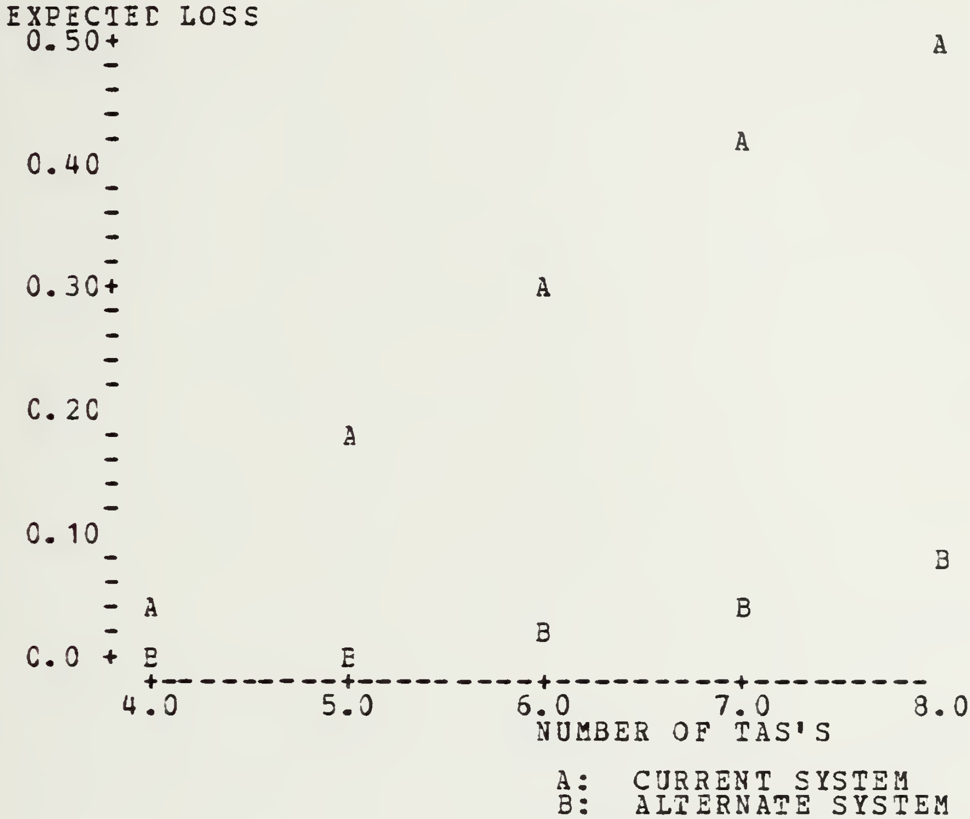


Figure D.17 System times with 190,818 characters.



REGRESSION EQUATION FOR A = - 0.398 +0.115 X1
S.D. ABOUT THE REGRESSION LINE = 0.01880

REGRESSION EQUATION FOR B = - 0.102 +0.0217 X1
S.D. ABOUT THE REGRESSION LINE = 0.01819

Figure D.18 Expected loss with 190,818 characters.

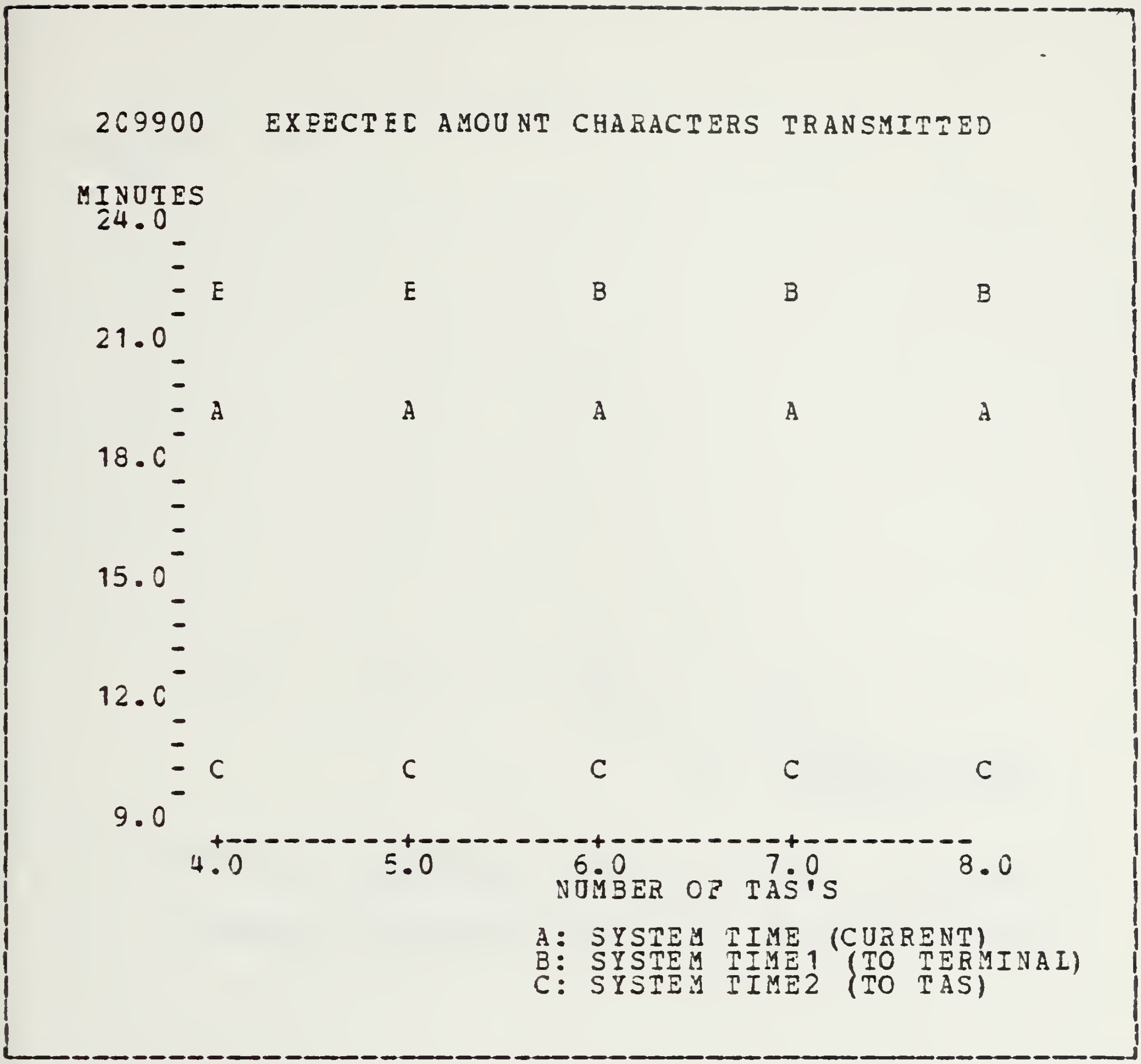
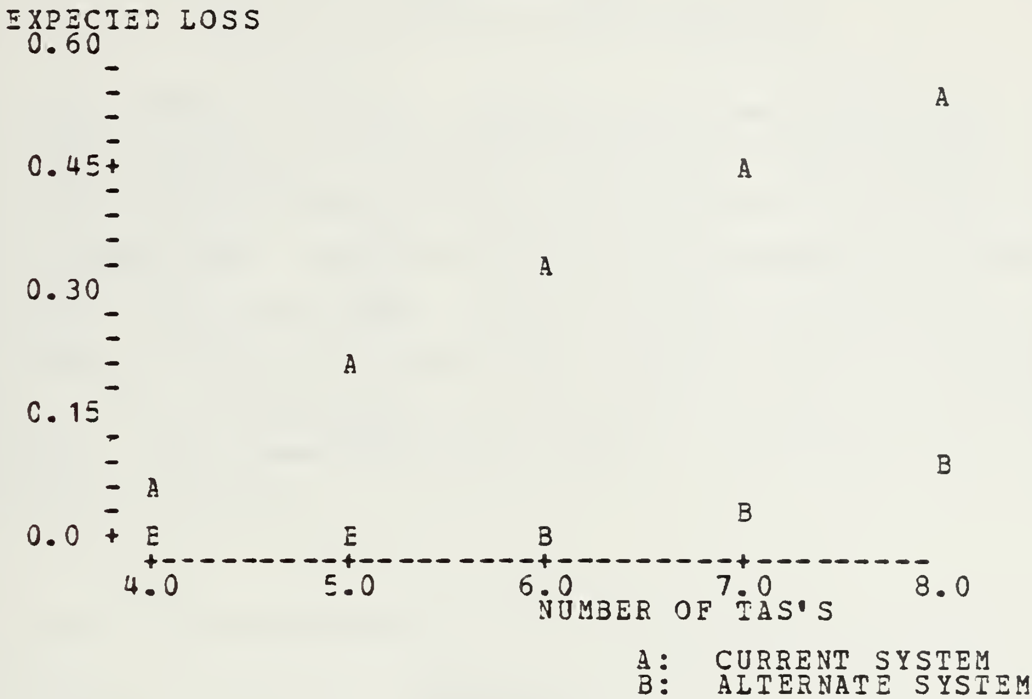


Figure D.19 System times with 209,900 characters.



REGRESSION EQUATION FOR A = - 0.384 +0.117 X1
S.D. ABOUT THE REGRESSION LINE = 0.02349

REGRESSION EQUATION FOR B = - 0.102 +0.0217 X1
S.D. ABOUT THE REGRESSION LINE = 0.01819

Figure D.20 Expected loss with 209,900 characters.

APPENDIX E

FIVE TO EIGHT TAS CONFIGURATION

These Tables are the results of simulation runs of a five TAS environment, where four TAS's are running in a fully operational mode and the arrival rate of the fifth TAS is increased over each run from 2.68 customers per hour to 17.68 customers per hour. For each of these arrival rates for TAS5, runs were made with varying the data transfer amount. The lambda column in the Tables refer to the lambda of TAS5. The remaining four TAS's are running at 17.68 customers per hour.

TABLE XXI

5 TAS, 89,018 characters: expected transfer amount

	current			alternate		
lambda	system time	exptd loss	prop. of use	system time1	system time2	exptd loss
2.68	12.33	.0	.61	13.18	8.05	.0
5.10	12.33	.006	.61	13.21	8.06	.0
5.61	12.32	.011	.61	13.21	8.07	.0
6.20	12.33	.009	.61	13.23	8.08	.0
7.47	12.32	.012	.61	13.22	8.08	.0
9.04	12.34	.015	.61	13.22	8.07	.0
12.02	12.33	.020	.61	13.23	8.10	.0
14.80	12.33	.031	.61	13.23	8.10	.001
17.68	12.33	.039	.61	13.24	8.11	.001

times are in minutes
lambda rate is per hour

TABLE XXII

5 TAS, 97,920 characters: expected transfer amount

	current			alternate		
lambda	system time	exptd loss	prop. of use	system time1	system time2	exptd loss
2.68	12.84	.007	.60	13.81	8.16	.0
5.10	12.85	.009	.60	13.84	8.18	.0
5.61	12.84	.014	.60	13.84	8.18	.0
6.20	12.85	.011	.60	13.87	8.20	.0
7.47	12.84	.016	.60	13.85	8.20	.0
9.04	12.85	.019	.60	13.85	8.20	.0
12.02	12.83	.025	.60	13.86	8.21	.0
14.80	12.82	.038	.60	13.87	8.22	.001
17.68	12.84	.049	.60	13.88	8.24	.001

times are in minutes
lambda rate is per hour

TABLE XXIII

5 TAS, 107,712 characters: expected transfer amount

	current			alternate		
lambda	system time	exptd loss	prop. of use	system time1	system time2	exptd loss
2.68	13.41	.011	.59	14.51	8.30	.0
5.10	13.42	.015	.59	14.54	8.32	.0
5.61	13.40	.018	.59	14.54	8.31	.0
6.20	13.42	.016	.59	14.57	8.34	.0
7.47	13.41	.022	.59	14.55	8.33	.0
9.04	13.42	.026	.59	14.56	8.34	.0
12.02	13.41	.033	.59	14.56	8.35	.0
14.80	13.38	.046	.59	14.58	8.37	.001
17.68	13.41	.058	.59	14.60	8.39	.001

times are in minutes
lambda rate is per hour

TABLE XXIV

5 TAS, 173,471 characters: expected transfer amount

	current			alternate		
lambda	system time	exptd loss	prop. of use	system time1	system time2	exptd loss
2.63	17.24	.052	.53	19.34	9.34	.0
5.10	17.18	.060	.53	19.38	9.36	.0
5.61	17.25	.066	.53	19.40	9.38	.0
6.20	17.19	.064	.53	19.44	9.40	.0
7.47	17.20	.078	.53	19.40	9.39	.0
9.04	17.24	.087	.53	19.43	9.41	.0
12.02	17.16	.104	.53	19.45	9.44	.0
14.80	17.21	.133	.53	19.49	9.49	.0
17.68	17.22	.151	.53	19.52	9.53	.0

times are in minutes
lambda rate is per hour

TABLE XXV

5 TAS, 190,818 characters: expected transfer amount

	current			alternate		
lambda	system time	exptd loss	prop. of use	system time1	system time2	exptd loss
2.63	18.18	.060	.52	20.66	9.65	.0
5.13	18.25	.080	.52	20.74	9.71	.0
5.61	18.22	.084	.52	20.72	9.70	.0
6.20	18.19	.082	.52	20.76	9.73	.0
7.47	18.21	.096	.52	20.72	9.72	.0
9.04	18.28	.110	.51	20.73	9.71	.0
12.02	18.22	.130	.52	20.77	9.77	.0
14.80	18.19	.157	.52	20.81	9.80	.001
17.68	18.18	.177	.52	20.87	9.88	.001

times are in minutes
lambda rate is per hour

APPENDIX F

EVENT LOGIC DIAGRAMS

This Appendix contains the logic diagrams of the internally generated events of the INS model. The following list of terms and definitions are included to aid in the reading of the diagrams.

1. THQUEUE: Queue of network requests. There is one for every possible combination of TAS and HOST. TASKs are placed in the appropriate THQUEUE as defined by the TAS and HOST identifier.
2. TASK: Temporary entity that may belong to a THQUEUE.
3. LQUEUE: High-speed facility queue. There is one for every possible combination of TAS and HOST. LTASKs are placed in the appropriate LQUEUE as defined by the TAS and HOST identifier.
4. LTASK: Temporary entity that may belong to a LQUEUE.
5. MU2: Expected service time for the interactive session when hard-copy demand is also requested.
6. MU1: Expected service time for an interactive session when no hard-copy demand is submitted.
7. MU31: Expected service time for data transfer in the current method.
8. MU32: Expected service time for the data transfer in the alternate method.
9. RETURN: Return to the SIMSCRIPT II.5 timing routine.

```
MAIN

         +--------------------------------+-
         | initialize model variables     |
         | read parameters                |
         +--------------------------------+-
                       |
                       |
+----------------------+-------------------------+-
|create permanent entities - paths between tas   |
| and host                                       |
|print network configuration                     |
+------------------------------------------------+-
                       |
+----------------------+--------------------------+-
|iterate through model for current method then   |
| for alternate method                           |
|                                                |
|    +----------------------------------------+  |
|    |initialize random number seeds          |  |
|    |schedule customer arrival to tas        |  |
|    |start simulation                        |  |
|    |at termination, print results           |  |
|    | reset model variables                  |  |
|    +----------------------------------------+  |
|                                                |
+------------------------------------------------+-
                       |
                       |
                    +---+-
                    |end|
                    +---+-
```

Figure F.1 MAIN.

```
TASx.ARRIVAL: custcmer arrival to a tas

+-----------------------------------------------------------+-
|schedule the next customer in amount of time               |
|defined by the exponential inter-arrival                   |
|distribution with parameter 1/lambda                       |
+-----------------------------------------------------------+-
                            |
                            |
                            |
          +-----------------+---------------
          |customer network request        |
          +--------------------------------
                            |
                            |
          yes               |      no
     +----------------------+--------------------+-
     |                                           |
     |                                           |
+---------------------------------------+        |
|1. determine database host             |        |
|2. determine work profile              |        |
+---------------------------------------+        |
                    |
                    |                         RETURN
                    |
       available    |     none available
     +--------------+----------------------------+-
     |                                           |
+--------------------------+   +--------------------------+-
|1. seize resource         |   |1.if queues permitted     |
|2. if hard-copy demand    |   | create temporary         |
| schedule event for       |   | task with customer       |
| customer to send         |   | attributes               |
| print cmd in amt of      |   | file task in appro-      |
| time defined by expo-    |   | priate thqueue           |
| nential distribution     |   |2. if no queues,          |
| with parameter mu2       |   | customer is lost to      |
|3. if no hard-copy        |   | the system               |
|demand,                   |   |                          |
| schedule customer        |   |                          |
| departure in amt of      |   |                          |
| time defined by expo-    |   |                          |
| nential distribution     |   |                          |
| with param mu1           |   |                          |
+--------------------------+   +--------------------------+-
            |                              |
            |                              |
          RETURN                         RETURN
```

Figure F.2 TAS ARRIVAL.

```
THDEPART:   Customer  departures for current method
and for cases when no hard-copy requests are made


        +------------------------------------+
        | update accounting variables        |
        | release network resources          |
        |                                    |
        +------------------------------------+
                          |
+-------------------------+-------------------------+
|If any other departures at this same time instant, |
| 1. remove this event from the events list         |
| 2. update accounting variables                    |
| 3. release network resources                      |
|                                                   |
+---------------------------------------------------+
                          |
                          |
+-------------------------+-------------------------+
|1. search through all non-empty thqueue's for a    |
| task that can be serviced by the available        |
| resources.                                        |
|                                                   |
|2. if such a task is found,                        |
|  remove task from thqueue                         |
|  seize the network resource                       |
|  if no hard-copy demands,                         |
|    schedule customer departure in amount of time  |
|    defined by exponential distribution with       |
|    parameter mu1                                  |
|  if hard-copy demand,                             |
|    schedule event to send print-command to tas    |
|    in amount of time defined by exponential       |
|    distribution with parameter mu2                |
|                                                   |
+---------------------------------------------------+
                          |
                          |
                        RETURN
```

Figure F.3   THDEPART.

USEND: Customer sends print command to TAS

schedule event at the user-tas to handle
user print command in the amount of time to
transmit command from terminal to cpu

RETURN

Figure F.4 USEND.

```
UC.ARRIVAL: At TAS to handle print command

          +----------------------------------+-
          |     state of tas resource        |
          |                                  |
          +----------------------------------+-
                           |
                           |
                           |
          busy             |                    free
      +--------------------+--------------------+-
      |                                         |
      |                                         |
  +-----------------------+   +-----------------------+-
  |1. create temporary    |   |1. set tas resource     |
  |entity, utask with     |   |busy                    |
  |customer attributes    |   |2. schedule event to    |
  |2. file stask in tas   |   |release tas resource    |
  | queue                 |   |in amount of time to    |
  |                       |   |process the             |
  |                       |   |request                 |
  +-----------------------+   +-----------------------+-
             |                           |
             |                           |
             |                           |
             |                           |
          RETURN                      RETURN
```

Figure F.5   UC.ARRIVAL.

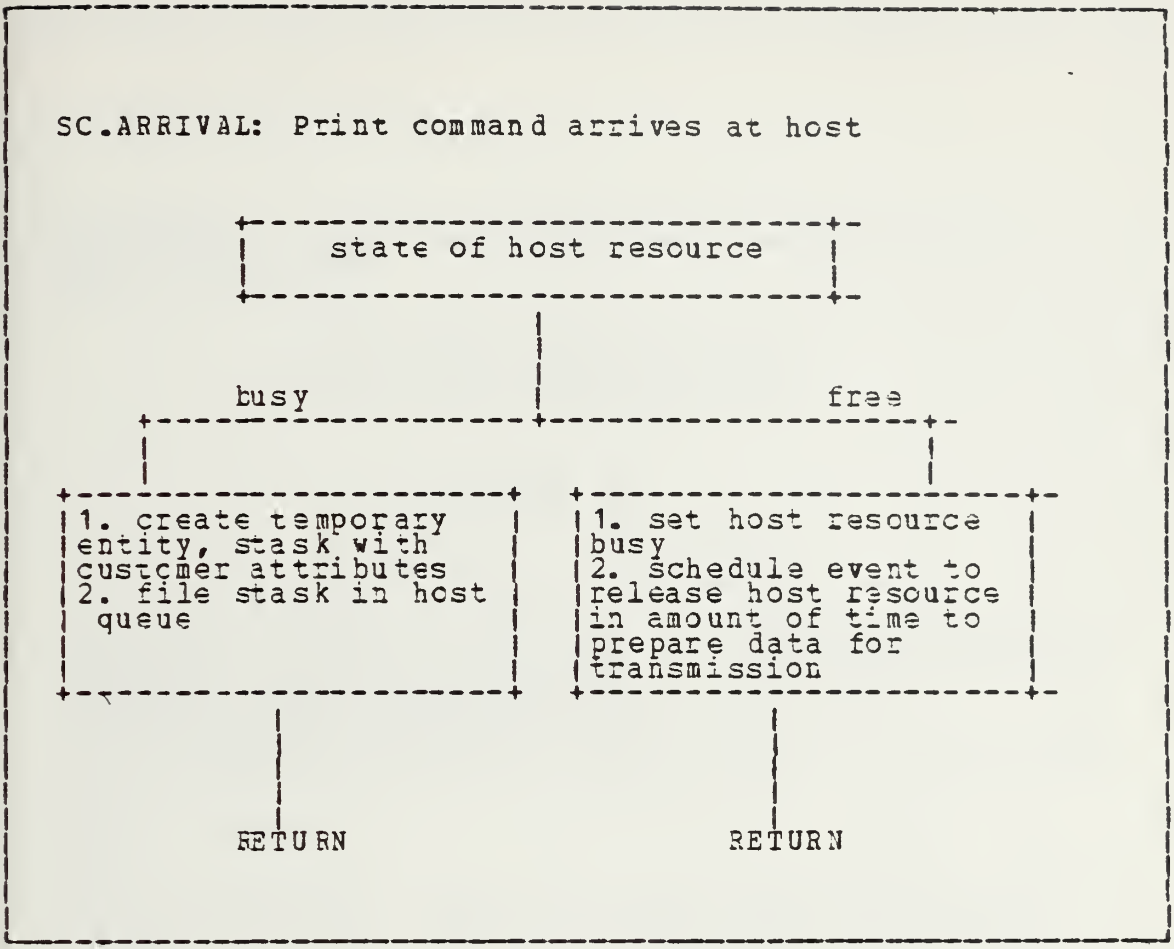


Figure F.6 SC.ARRIVAL.

SC.DEPART: host has completed preparation of data for transfer

```
                  +-------------------------+-
                  | set host resource free |
                  +-------------------------+-
                                |
                                |
                         +-----+---+-
                         | method  |
                         +---------+-
                                |
                      current   |   alternate
              +-----------------+------------------+-
              |                                    |
+--------------------------------+    +-----------------+-
|schedule a completion of data  |    |release network    |
|transfer in amt of time de-    |    |resources          |
|fined by the exponential dis-  |    |                   |
|tribution with parameter mu31  |    |                   |
+--------------------------------+    +-----------------+-
              |                                |
+--+          |                                |
|a |----------|                                |
+--+          |                                |
              |                                |
+----------+---+       +--------------------+-------+-
|state of host |       |state of high speed facility|
|queue         |       |                            |
+--------------+       +----------------------------+-
   |                                  |
   |                          busy    |   free
   |          +-----------------------------+-
   |          |                             |
   |   +----------------------+   +-------------------------+-
   |   |1. create temporary  |   |1. seize high speed       |
   |   | ltask with cus-     |   |resource                  |
   |   | tomer attributes    |   |2. schedule comple-       |
   |   |2. file ltask in     |   | tion of data xfer in     |
   |   | lqueue              |   | amt of time defined      |
   |   |                     |   | by the exponential       |
   |   |                     |   | distribution with        |
   |   |                     |   | parameter mu32           |
   |   +----------------------+   +-------------------------+-
   |            |                           |
+-+-+        +-+-+                       +-+-+-
|z2 |        |z1 |                       |z1 |
+---+        +---+                       +---+-
```

Figure F.7 SC.DEPART.

```
SC.DEPART (continuation 1)

                    +----+-
                    | z1 |
                    +----+-
                       |
                       |
           +-----------+-----------+-
           |     state of thqueue  |
           +-----------------------+-
                       |
         not empty     |    empty
     +-----------------+------------------------+-
     |                                          |
     |                                          |
+---------------------------------------+       |
|1. remove earliest task in thqueue that|       |
| can be serviced by available resources|       |
|2. seize network resources             |       |
|3. if nc hard-copy demands,            |       |
| schedule customer departure in amt    |       |
| of time defined by exponential        |       |
| distribution with parameter mu1       |       |
|4. if hard-copy demands,               |       |
| schedule event tc send print command  |       |
| tc tas in amount of time defined by   |       |
| exponential distribution with         |       |
| parameter mu2.                        |       |
|                                       |       |
+---------------------------------------+       |
                  |                             |
                  |                             |
                  |                             |
                +-+-+                         +-+-+-
                | a |                         | a |
                +---+                         +---+-
```

Figure F.8   SC.DEPART (continuation 1).

```
SC.DEPART (continuation 2)

                      +-----+-
                      | z2  |
                      +-----+-
                         |
                         |
          not empty      |    empty
     +-------------------+----------------------------+-
     |                                                |
     |                                                |
+-----------------------------------------------+     |
|1. remove earliest stask from queue            |     |
|2. set host resource busy                      |     |
|3. schedule this event to release host         |     |
| resource in amount of time to                 |     |
| prepare data for transfer.                    |     |
|                                               |     |
+-----------------------------------------------+     |
                    |                                 |
                    |                                 |
                 RETURN                            RETURN
```

Figure F.9 SC.DEPART (continuation 2).

```
+---------------------------------------------------------------------------+
|                                                                           |
| LTHDEPART: hard-copy transfer completed                                   |
|                                                                           |
|          +------------------------------+-                                |
|          |update accounting variables   |                                |
|          +------------------------------+-                                |
|                         |                                                |
|                         |                                                |
|                   +-----+----+-                                          |
|                   | method   |                                           |
|                   +----------+-                                          |
|                         |                                                |
|              current    | alternate                                      |
|       +-----------------+-----------+-                                   |
|       |                             |                                    |
| +-----------------+ +------------------------------+-                     |
| |schedule customer| |state of high speed queue|                          |
| |departure        | |                              |                    |
| +-----------------+ +------------------------------+-                     |
|   |                               |                                       |
|   |                  not empty    |   empty                               |
|   |       +----------------------+--------+-                              |
|   |       |                               |                               |
|   |    +------------------------------------+                             |
|   |    |1. remove earliest ltask from      |  |                           |
|   |    | lqueue                             |  |                          |
|   |    |2. schedule this event in amt of   |  |                           |
|   |    | time defined by exponential       |  |                           |
|   |    | distribution with parameter mu32  |  |                           |
|   |    |3. destroy ltask                   |  |                           |
|   |    +------------------------------------+  |                          |
|   |                  |                         |                          |
|   |                  |                         |                          |
| +-+------------------+-------------------------+------+-                  |
| |if any other transfers completed at this time        |                   |
| |1. remove event from events-list                     |                   |
| |2. update accounting variables                       |                   |
| |3. for alternate case,                               |                   |
| | if state of hi-speed queue not-empty,               |                   |
| |  remove earliest ltask from lqueue                  |                   |
| |  schedule this event in amt of time defined         |                   |
| |  by exponential distribution with parameter mu32    |                   |
| |  destroy ltask                                      |                   |
| |4. for current case, schedule a customer             |                   |
| |  departure                                          |                   |
| +-----------------------------------------------------+-                  |
|                      |                                                    |
|                      |                                                    |
|                   RETURN                                                 |
|                                                                           |
+---------------------------------------------------------------------------+
```

Figure F.10 LTHDEPART.

```
CLOSING: Halts customer arrivals to the TASs

+-------------------------------------------------------+-
|                                                       |
|     Cancel the customer arrival to the TAS            |
|                                                       |
+-------------------------------------------------------+-
                          |
                          |
                       RETURN
```

Figure F.11 CLOSING.

APPENDIX G

INS PROGRAM LISTING

This Appendix contains the program listing for the simulation model and the Job Control Language statements that were used to run the simulation at the W. R. Church Computer Center.

```
//KIM JOE   (2241,1230),'JB KIM',CLASS=B
//*MAIN ORG=NPGVM1.2241P
// EXEC SIM25CLG
//SIM.SYSIN  DD *
''*****************************************************************
''  12 MAY 83: GOT DATA FROM COINS. MAKING CHANGES FOR
''  CHECK IF ONLY INTERACTIVE AND NO PRINT COMMAND...THEN 1 DISTRIBUT
''  FOR INTERACTIVE TIME;
''  OTHERWISE, USE DIFFERENT DISTRIBUTION FOR SEARCH WITH PRINT COMM.
''
''*****************************************************************


PREAMBLE
      DEFINE KNTR, CASE AS INTEGER VARIABLES
      DEFINE SOLIS, NUIS, USIS AS  VARIABLES
      DEFINE CCASE AS ALPHA  VARIABLES
      DEFINE MATRIX AS REAL  2-DIMENSIONAL ARRAY
      DEFINE TSPD.FLAG AS INTEGER 1-DIMENSIONAL  ARRAY
      DEFINE HSPD.FLAG AS INTEGER 1-DIMENSIONAL  ARRAY


   PERMANENT ENTITIES

         EVERY TAS HAS A TAS.MAX, A TNUM, A S1.FLAG,A TPATH,
            A UBUSY,
            A UWAIT.TIME,
            A UFULL.TIME
            AND OWNS A UQUEUE
            DEFINE TAS.MAX, TNUM, S1.FLAG, UBUSY, TPATH
               AS INTEGER VARIABLES
            DEFINE UWAIT.TIME, UFULL.TIME AS VARIABLES

         EVERY HOST HAS A HOST.MAX, A HNUM, A S2.FLAG, A HPATH,
         A SBUSY,
         A SWAIT.TIME,
         A SFULL.TIME
         AND OWNS A SQUEUE
            DEFINE  HOST.MAX, HNUM,
               S2.FLAG, HPATH AS INTEGER VARIABLES
            DEFINE  SWAIT.TIME, SFULL.TIME
                AS VARIABLES

         EVERY TASHOST OWNS A THQUEUE AND HAS
            A WORK,
            A USER,
            A SERVER,
```

```
          A THBUSY,
          A FLOW.CAPACITY
          DEFINE USER, SERVER                      AS INTEGER VARIABLES
          DEFINE THBUSY, FLOW.CAPACITY             AS INTEGER VARIABLES

       EVERY LPATH OWNS A LQUEUE AND HAS
          A RECEIVER,
          A SENDER,
          A BUSY
          DEFINE BUSY, SENDER, RECEIVER AS INTEGER VARIABLES


TEMPORARY ENTITIES

       EVERY TASK HAS A TAS.CHOICE,
          A HOST.CHOICE,
          AN ARRIVAL.TIME,
          A INTER,
          A START.TIME,
          A LLONG,
          AND MAY BELONG TO A THQUEUE
          DEFINE ARRIVAL.TIME, LLONG, INTER, START.TIME AS VARIABLES
          DEFINE TAS.CHOICE, HOST.CHOICE, LONG AS INTEGER VARIABLES

EVERY UTASK HAS A UARR.TIME,
   A TTU,
   A UTTHH,
   A UC.TAS,
   A UC.CTIME,
   A UC.HOST,
   A UC.AMT,
   A UC.RES
   AND MAY BELONG TO A UQUEUE
   DEFINE UTTHH, UC.TAS, UC.HOST AS INTEGER VARIABLES
   DEFINE UARR.TIME, TTU, UC.CTIME, UC.AMT, UC.RES AS VARIABLES

EVERY STASK HAS A SARR.TIME,
   A TTS,
   A TTHH,
   A TTT,
   A TTH,
   A SC.AMT,
   A SC.RES
   AND MAY BELONG TO A SQUEUE
   DEFINE TTHH, TTT, TTH AS INTEGER VARIABLES
```

```
   DEFINE SARR.TIME, TTS, SC.AMT, SC.RES AS VARIABLES

EVERY ITASK HAS A LARR.TIME,
   A TAKER,
   A GIVER,
   A LENGTH, A LIN.TIME,
   A RES.TIME
   AND MAY BELONG TO A LQUEUE
   DEFINE TAKER, GIVER AS INTEGER VARIABLES
   DEFINE LARR.TIME, LENGTH, RES.TIME AS VARIABLES
   DEFINE LIN.TIME AS A VARIABLE

EVENT NOTICES INCLUDE TAS1.ARRIVAL,
      TAS2.ARRIVAL,
      TAS3.ARRIVAL,
      TAS4.ARRIVAL,
      TAS5.ARRIVAL,
      TAS6.ARRIVAL,
      TAS7.ARRIVAL,
      TAS8.ARRIVAL,
      CLOSING

EVERY THDEPART HAS  A TTOHOST,
      A TASS,
      A HOSTT,
      A DATA.AMT
   DEFINE TTOHOST, TASS, HOSTT AS INTEGER VARIABLES
   DEFINE DATA.AMT AS A VARIABLE

EVERY USEND HAS A US.TIME,
   A UTAS,
   A UHOST,
   A UTAS.HOST,
   A ULONG.AMT
   DEFINE UTAS, UHOST, UTAS.HOST AS INTEGER VARIABLES
   DEFINE US.TIME, ULONG.AMT AS VARIABLES

EVERY UC.ARRIVAL HAS A UCTAS.HOST,
   A UCTAS,
   A UCHOST,
   A UCLONG.AMT,
   A FCMD.TIME,
   A U1RES.TIME
   DEFINE UCTAS.HOST, UCTAS, UCHOST AS INTEGER VARIABLES
   DEFINE UCLONG.AMT, U1RES.TIME, FCMD.TIME AS VARIABLES

EVERY UC.DEPART HAS A DEPTH,
```

```
    A DEPT,
    A DEPH,
    A DEP.LONG.AMT,
    A FCTIME,
    A U2RES.TIME
    DEFINE DEPTH, DEPT, DEPH AS INTEGER VARIABLES
    DEFINE DEP.LONG.AMT, U2RES.TIME, FCTIME AS VARIABLES

EVERY SC.ARRIVAL HAS A STAS.HOST,
    A STT,
    A SHH,
    A SLONG.AMT,
    A S1RES.TIME
    DEFINE STAS.HOST, STT, SHH AS INTEGER VARIABLES
    DEFINE SLONG.AMT, S1RES.TIME AS VARIABLES

EVERY SC.DEPART HAS A SDTAS.HOST,
    A SDT,
    A SDH,
    A SDLONG.AMT,
    A S2RES.TIME
    DEFINE SDTAS.HOST, SDT, SDH AS INTEGER VARIABLES
    DEFINE SDLONG.AMT, S2RES.TIME AS VARIABLES

EVERY LTHDEPART HAS A WWAY,
    A WTAS,
    A WHOST,
    A WDATA.AMT,
    A WRES.TIME
    DEFINE WWAY, WTAS, WHOST AS INTEGER VARIABLES
    DEFINE WDATA.AMT, WRES.TIME  AS VARIABLES

PRIORITY ORDER IS THDEPART,
      LTHDEPART,
      SC.DEPART,
      SC.ARRIVAL,
      UC.DEPART,
      UC.ARRIVAL,
      USEND,
      TAS1.ARRIVAL,
      TAS2.ARRIVAL,
      TAS3.ARRIVAL,
      TAS4.ARRIVAL,
      TAS5.ARRIVAL,
      TAS6.ARRIVAL,
      TAS7.ARRIVAL,
      TAS8.ARRIVAL,
```

```
CLOSING

DEFINE SECONDS TO MEAN UNITS
DEFINE YES TO MEAN 1
DEFINE MINUTES TO MEAN 60 * UNITS
DEFINE HOURS TO MEAN 60 * 60 * UNITS
DEFINE OCCUPIED TO MEAN 1
DEFINE FREE TO MEAN 0
DEFINE CURRENT TO MEAN 1
DEFINE ALTERNATE TO MEAN 2

     DEFINE SEED AS INTEGER, 1-DIMENSIONAL ARRAY
     DEFINE  LAMBDA, TSTUFF AS REAL 1-DIMENSIONAL ARRAY
     DEFINE T.HOST AS REAL 2-DIMENSIONAL ARRAY
     DEFINE MO,DAY, HR, MINU AS INTEGER VARIABLES
     DEFINE LONG.FLAG AS A VARIABLE
     DEFINE MU AS VARIABLES
     DEFINE ERR, CONFIG, I,J AS INTEGER VARIABLES
     DEFINE TMAX, HMAX AS INTEGER, 1-DIMENSIONAL ARRAYS
     DEFINE TASNUM, HOSTNUM, STASNUM AS INTEGER VARIABLES
     DEFINE COMPLETED, NOWAIT,QWAIT AS  INTEGER VARIABLES
     DEFINE LCOMPLETED, LNOWAIT, LQWAIT AS INTEGER VARIABLES
     DEFINE TURN.AWAY, REFUSED AS INTEGER VARIABLES

     TALLY A.HPATH AS THE AVERAGE                      OF HPATH
     ACCUMULATE HPLAIN.USE AS THE AVERAGE              OF N.HOST
     TALLY UTILIZATION AS THE AVERAGE,
           IRHISTO(0.0 TO 1.0 BY 0.1) AS THE HISTOGRAM
                                                       OF USAGE
     TALLY TUTILIZATION AS THE AVERAGE,
           TRHISTO(0.0 TO 1.0 BY 0.1) AS THE HISTOGRAM
                                                       OF TUSAGE
     TALLY A.TPATH AS THE AVERAGE                      OF TPATH
     TALLY AVG.TH.Q AS THE AVERAGE                     OF N.THQUEUE
     TALLY AV.PATH.Q AS THE AVERAGE                    OF N.LQUEUE
     TALLY PATH.USE AS THE AVERAGE                     OF BUSY
     TALLY LAFULL AS THE AVERAGE,
           LFHISTO(0 TO 2520 BY 180) AS THE HISTOGRAM,
           INFULL AS THE NUMBER                        OF IFULL.TIME
     TALLY XAFULL AS THE AVERAGE,
           XNFULL AS THE NUMBER                         OF XFULL.TIME
     TALLY PDONE AS THE NUMBER,
           MU.LWAIT AS THE AVERAGE                     OF LWAIT.TIME
     TALLY LNUM.QUE AS THE NUMBER,
           LAVG.QUE AS THE AVERAGE                     OF LQUE.WAIT
     TALLY THUSE AS THE AVERAGE                        OF THBUSY
```

```
TALLY INQUE.AVG AS THE AVERAGE                  OF INQUEUE
TALLY ANOT.EMP.Q AS THE AVERAGE                 OF NOT.EMP.Q
TALLY LINQ.AVG AS THE AVERAGE                   OF LINQUE
TALLY NO.DONE AS THE NUMBER,
      WHISTO(0 TO 2520 BY 180) AS THE HISTOGRAM,
      MEAN.WAIT AS THE AVERAGE                  OF WAIT.TIME
TALLY WK.NUM AS THE NUMBER,
      WK.AVG AS THE AVERAGE                     OF DURATION
TALLY NUM.QUE AS THE NUMBER,
      QHISTO(0 TO 2520 BY 180) AS THE HISTOGRAM,
      AVG.QUE AS THE AVERAGE                    OF QUE.WAIT.TIME
TALLY NUMFUL AS THE NUMBER,
      FHISTO(0 TO 2520 BY 180) AS THE HISTOGRAM,
      AVG.FULL AS THE AVERAGE                   OF FULL.TIME
TALLY L1AVGWAIT AS THE AVERAGE,
      L1NUMWAIT AS THE NUMBER                   OF L1WAIT.TIME
TALLY L1AVFULL AS THE AVERAGE,
      L1FULL AS THE NUMBER                      OF L1FULL.TIME
TALLY L2AVGWAIT AS THE AVERAGE,
      L2NUMWAIT AS THE NUMBER                   OF L2WAIT.TIME
TALLY L2AVFULL AS THE AVERAGE,
      L2FULL AS THE NUMBER                      OF L2FULL.TIME
TALLY LAVG AS THE AVERAGE,
      LNUM AS THE NUMBER                        OF LRES.TIME
DEFINE INQUEUE AS AN INTEGER VARIABLE
DEFINE NOT.EMP.Q,
      LINQUE                                    AS INTEGER VARIABLES
DEFINE L1WAIT.TIME, L1FULL.TIME,
       L2WAIT.TIME, L2FULL.TIME,
       LWAIT.TIME, LQUE.WAIT, LRES.TIME,
       FULL.TIME, DURATION, QUE.WAIT.TIME, WAIT.TIME,
       USAGE, TUSAGE,
       RATE1, RATE2, LFULL.TIME,
       XFULL.TIME
       AS VARIABLES

END
```

```
''*****************************************************************
''                            MAIN
''*****************************************************************

MAIN

   RESERVE SEED(*) AS 38


      PERFORM INITIALIZATION
      RELEASE INITIALIZATION
      IF ERR = 1,
         GO ERR.1
      ELSE
      IF ERR = 2,
         GO ERR.2
      ELSE
         RELEASE SEED.V(*)
         RESERVE SEED.V(*) AS 38
         LET N.TASHOST = N.TAS * N.HOST
         CREATE EVERY TASHOST

         LET N.LPATH = N.TASHOST
         CREATE EVERY LPATH
         PERFORM DEBUG.LIST
'REDC'
      FOR I = 1 TO 38
         DO
            LET SEED.V(I) = SEED(I)
         LOOP
      PERFORM SET.UP
      PERFORM HEADER
            FOR I = 1 TO 38
               DO
               LET SEED.V(I) = SEED(I)
               LOOP

      SCHEDULE A TAS1.ARRIVAL NOW
      SCHEDULE A TAS2.ARRIVAL NOW
      SCHEDULE A TAS3.ARRIVAL NOW
      SCHEDULE A TAS4.ARRIVAL NOW
''    SCHEDULE A TAS5.ARRIVAL NOW
''    SCHEDULE A TAS6.ARRIVAL NOW
''    SCHEDULE A TAS7.ARRIVAL NOW
''    SCHEDULE A TAS8.ARRIVAL NOW
      SCHEDULE A CLOSING IN 001*3600 SECONDS
```

```
        START SIMULATICN

        PERFORM RITE1
        PERFORM AAGAIN
        IF TURN.AWAY = 1,
             LET TURN.AWAY = 0
             GO REDO

        REGARDLESS

        IF CASE = 1,
           PERFORM COMPARE
           GO REDO
        ELSE

'CMORE'
        PERFORM AGAIN
        GO REDO


'ERR.1'
     SKIP 2 OUTPUT LINES
     PRINT 2 LINES WITH TASNUM, HOSTNUM, STASNUM THUS
     ILIEGAL CONFIGURATION SUBMITTED, THIS IS WHAT I RECEIVED:
      # TAS'S = **    # HOST'S =  **    # STAS'S = **

     GO PAU

'ERR.2'
     SKIP 2 OUTPUT LINES
     PRINT 3 LINES THUS

     ONE OF THE SERVER-TAS'S HAS 1 OR LESS PORTS ASSIGNED
      S-TAS    PORTS

     FOR I = TASNUM TO STASNUM +TASNUM ,
        DO
        PRINT 1 LINE WITH I - TASNUM, TMAX(I)THUS
          **     **
        LOOP
     GO PAU
'PAU'

END
```

```
''***********************************************************************
''                            INITIALIZATION ROUTINE
''***********************************************************************

ROUTINE FOR INITIALIZATION

''----------------------------------------------------------------------
''   MATRIX(I,J) :
''      I:   EACH  ROW  CONTAINS DISTRIBUTION INFORMATION FCR HOST I
''      J:   1 TO  10 DEFINE THE INDIVIDUAL DISTRIBUTION PARAMS AS--
''           COL:  1  PERCENTAGE INTERACTIVE ONLY
''           COL:  2 FOR INTER ONLY, WHAT DISTRIBUTION
''           COL:  3,4  PARAMETERS FOR THE DISTRIBUTION, (MU AND S.D.)
''           COL:  5  FCR BOTH, DISTRIBUTION FOR INTER
''           COL:  6,7  PARAMETERS FOR THE DISTRIBUTION IN COL 5
''           COL:  8  FOR BOTH, DISTRIBUTION FOR LONG.XFER
''           COL:  9,10  PARAMETERS FOR THE DISTRIBUTION IN COL 8
''           COL:  11 - END PERCENTAGE OF WORK ORIGINATING FROM
''                 EACH TAS(K)   GOING TO HOST(I)
''----------------------------------------------------------------------

   DEFINE I,J,K, FLAG, TIP, STEP  AS INTEGER VARIABLES

   DEFINE XMATRIX AS REAL 2-DIMENSIONAL ARRAY


      LET RATE1 = 2400.
      LET RATE2 = 20000.
      FOR I = 1 TO 10
         DO
            LET SEED(I) = SEED.V(I)
         LOOP
      FOR KNTR = 11 TO 38,
         LET SEED(KNTR) = RANDI.F(2011,31737,10)


      LET FLAG = 0
      LET ERR = 0
      READ TASNUM, HCSTNUM, STASNUM
      LET CASE = 1
      RESERVE XMATRIX(*,*) AS HCSTNUM+STASNUM BY 10
      FOR I = 1 TO HCSTNUM+STASNUM,
         FOR K = 1 TO 10, READ XMATRIX(I,K)
PRINT 1 LINE WITH CASE  THUS
  THIS IS THE VALUE CF CASE   *****
   IF TASNUM > 0,
```

```
RESERVE TMAX(*), TSPD.FLAG(*) AS TASNUM + STASNUM
RESERVE LAMBDA(*), TSTUFF(*) AS TASNUM + STASNUM
RESERVE T.HOST(*,*) AS TASNUM + STASNUM BY 3
FOR I = 1 TO TASNUM
   DO
      READ LAMBDA(I), TMAX(I),TSTUFF(I),
         T.HOST(I,1), T.HOST(I,2), T.HOST(I,3), TSPD.FLAG(I)
      IF CASE = ALTERNATE AND TSPD.FLAG(I) = YES,
         SUBTRACT 1 FROM TMAX(I)
      REGARDLESS
   LOOP
ALWAYS

IF HOSTNUM > 0,
   RESERVE HMAX(*), HSPD.FLAG(*) AS HOSTNUM + STASNUM

   FOR I = 1 TO HOSTNUM
   DO
      READ HMAX(I), HSPD.FLAG(I)
         IF CASE = ALTERNATE AND HSPD.FLAG(I) = YES,
            SUBTRACT 1 FROM HMAX(I)
         REGARDLESS
   LOOP
ALWAYS

IF STASNUM > 0,
   LET I = TASNUM
   LET J = HOSTNUM
   LET FLAG = 1

   FOR K = 1 TO STASNUM
   DO
      ADD 1 TO I
      ADD 1 TO J
      READ LAMBDA(I),TMAX(I), TSTUFF(I)
      READ T.HOST(I,1), T.HOST(I,2), T.HOST(I,3)
      READ TSPD.FLAG(I)
      IF TMAX(I) LE 0,  LET ERR = 2
         RETURN
      ALWAYS

      IF CASE = ALTERNATE AND TSPD.FLAG(I) = YES,
         SUBTRACT 1 FROM TMAX(I)
      REGARDLESS
      LET HMAX(J) = TMAX(I)
      LET HSPD.FLAG(J) = TSPD.FLAG(I)
   LOOP
```

```
    LET N.TAS = TASNUM + STASNUM
    LET N.HOST = HOSTNUM + STASNUM
ELSE
    LET N.TAS = TASNUM
    LET N.HOST = HOSTNUM
REGARDLESS
IF N.TAS = 0 OR N.HOST = 0,
    LET ERR = 1
ELSE

    CREATE EVERY TAS
    LET I = 0
    LET J = 0
    LET TIP = N.TAS - STASNUM + 1
    FOR EACH TAS
        DO
            ADD 1 TO I
            LET TAS.MAX(TAS) = TMAX(I)
            IF I >= TIP,
                ADD 1 TO J
                LET S1.FLAG(TAS) = J
            ALWAYS
        LOOP
    CREATE EVERY HOST
    RESERVE MATRIX(*,*) AS N.HOST BY 10
    LET I = 0
    LET J = 0
    LET TIP = N.HOST - STASNUM + 1
    FOR EACH HOST
        DO
            ADD 1 TO I
            LET HOST.MAX(HOST) = HMAX(I)
            FOR STEP = 1 TO 10,
                LET MATRIX(HOST,STEP) = XMATRIX(HOST,STEP)
            IF I >= TIP,
                ADD 1 TO J
                LET S2.FLAG(HOST) = J
            ALWAYS
        LOOP
    LET ERR = 0
REGARDLESS
RELEASE XMATRIX(*,*)
    RETURN
    END
```

```
''*************************************************************************
''                              SET.UP ROUTINE
''*************************************************************************

ROUTINE FOR SET.UP

DEFINE USE, SERVE AS INTEGER VARIABLES
DEFINE I AS AN INTEGER VARIABLE


   LET I = 0
   FOR USE = 1 TO N.TAS
      DO
      FOR SERVE = 1 TC N.HOST
         DO
         ADD 1 TO I
         LET USER(I) = USE
         LET SERVER(I) = SERVE
         LET FLOW.CAPACITY(I) = MIN.F(TAS.MAX(USE), HOST.MAX(SERVE) )
         LOOP
      LOOP
      LET CCNFIG = 1


   RETURN

END
```

```
''***************   LIST DEBUG ROUTINE
''
ROUTINE FOR DEBUG.LIST

   PRINT 2 LINE THUS
HOST  INX  I-D   P1  P2     L-DIS    P1        P2    ILD    P1  FG   MX


FOR K = 1 TC N.HOST,
  PRINT 1 LINE WITH K, MATRIX(K,1),
   MATRIX(K,2),MATRIX(K,3),
   MATRIX(K,4),MATRIX(K,5),
   MATRIX(K,6),MATRIX(K,7),
   MATRIX(K,8),MATRIX(K,9), MATRIX(K,10),
   HSPD.FLAG(K), HOST.MAX(K)
   THUS
*    *.**  **   *.**  *.**   **     **.*    **.*     **  ***.*   **.* * *

  SKIP 1 OUTPUT LINE
  PRINT 1 LINE THUS
TAS   LAMBDA  TMAX    INTER%     -SOLIS      -USIS    -NUIS        SPD.F
FOR K = 1 TC N.TAS,
  PRINT 1 LINE WITH K,
            LAMBDA(K), TAS.MAX(K), TSTUFF(K), T.HOST(K,1),
            T.HOST(K,2), T.HOST(K,3), TSPD.FLAG(K)
   THUS
 ***  **.**   ****     *.**      *.***     *.***     *.***      ***

      SKIP 1 OUTPUT LINE
      PRINT 1 LINE WITH MO,DAY,HR,MINU,TASNUM, HOSTNUM, STASNUM THUS
 *****  **   AT **:**    ***  TAS'S    ***   HOSTS      *** S-TAS'S

END
```

```
''*************************************************************************
''                          SEE.SET ROUTINE
''*************************************************************************

ROUTINE FOR SEE.SET

      SKIP 2 OUTPUT LINES
      PRINT 2 LINES THUS
      PATH   USER(T)   SERVER(H)    FLOW.CAPACITY

      FOR EACH TASHOST
      PRINT 1 LINE WITH TASHOST, USER(TASHOST), SERVER(TASHOST),
      FLOW.CAPACITY(TASHOST)THUS
     **        **          **               ***

   PRINT 3  LINES THUS

   TAS   S1.F   MAX

   FOR EACH TAS
   PRINT 1 LINE WITH TAS, S1.FLAG(TAS),  TAS.MAX(TAS) THUS
    **   (**)**

   PRINT 3  LINES THUS

   HOST  S2.F   MAX

   FOR EACH HOST
   PRINT 1 LINE WITH HOST,S2.FLAG(HOST), HOST.MAX(HOST) THUS
    **   (**)    **

END
```

```
''*************************************************************************
''                         SET UP COMPARISON RUN
''*************************************************************************

ROUTINE FCR COMPARE

''   GCING FROM CURRENT CONFIGURATON, TO ALTERNATE PROPOSAL
''   AIL PARAMETERS REMAIN THE SAME EXCEPT THAT THE MAX-VALUES
''   MUST BE MODIFIED (SUBTRACT 1)

LET CASE = 2
LET TURN.AWAY = 1
FOR EVERY HCST,
   DC
      IF HSPD.FLAG(HCST) = YES,
         SUETRACT 1 FROM HOST.MAX(HOST)
      REGARDLESS
   LCCP

FOR EVERY TAS,
   DO
      IF TSPD.FLAG(TAS) = YES,
         SUETRACT 1 FROM TAS.MAX(TAS)
      REGARDLESS
   LOCP

PERFORM SET.UP

PRINT 3 LINES THUS
================--------------------------------------------------------
                      READY TO DO ALTERNATE PROPOSAL
----------------========================================================


      PERFORM DEBUG.LIST

RETURN
END
```

```
''*********************************************************************
''                            HEADER ROUTINE
''*********************************************************************

ROUTINE FOR HEADER

   SKIP 1 OUTPUT LINE
   IF TURN.AWAY = 1,
      GO TO NOW1 OR ALT1 PER CASE

'NOW1'
      PRINT 1 LINES THUS
UTILI    CUSTT    AVG.SYS.TIM    REFUSED

GO OUT
'ALT1'

      PRINT 2 LINES THUS
UTILI    CUSTT    AVG.SYS.TIM    AVG. SYS.TIM   # XFERS    REFUSED
                  (TO TERM)      (TO CPU-      )  WAITED

GO OUT
   ELSE
      GO TO NOW2 OR ALT2 PER CASE

'NOW2'
      PRINT 1 LINES THUS
UTILI  CUSTT   AVG. WAIT  AVG.Q.QWAIT  AVG.SYS.TIM  WAITED   AVG.QUE

GO OUT
'ALT2'

      PRINT 2 LINES THUS
UTILI  CUSTT  AVG.Q.WAIT  A.SYS.TIME  AVG.SYS.TIM  WAITED   AVG.QUE
                           (TO TERM)  (TO CPU-       )

GO OUT

'OUT'
   SKIP 1 OUTPUT LINES
      RETURN
END
```

```
''**************************************************************************
''                              WRITER ROUTINE
''**************************************************************************

ROUTINE FCR WRITER

IF TURN.AWAY = 1,

   PRINT 1 LINE WITH
      100.0 - VARY.PROB,
      NO.DONE,
      AVG.FULL/60.0,
      LAVG.QUE/60.0,
      LNUM.QUE,
      REFUSED
      THUS
 ***.*   *****     ****.**      ****.**     ****      ****

ELSE

   PRINT 1 LINE WITH
   100.0 - VARY.PROB,
   NO.DCNE,
   MEAN.WAIT/60.,
   AVG.QUE/60.,
   AVG.FULL/60.,
   NUM.QUE,
   INQUE.AVG
   THUS
 ***.*  *****  ****.**    *****.**   *****.**      ****     *****

ALWAYS

RETURN
END
```

```
''*************************************************************************
''                                 ROUTINE FOR RITE1
''*************************************************************************

ROUTINE FOR RITE1
IF TURN.AWAY = 1,

   GO TC NCW1 OR ALT1 PER CASE

'NOW1'

      PRINT 1 LINE WITH
         UTILIZATION,
         NO.DONE,
         AVG.FULL/60.0,
         REFUSED/(REFUSED + NO.DONE),
         LAVG/60.,
         LNUM
         THUS
 **.**   *****     ****.**     .******  *****.*** LONG ****L.REQS

      GO NEXTR

'ALT1'

      PRINT 4 LINE WITH
         UTILIZATION,
         NO.DONE,
         AVG.FULL/60.0,
         LAFULL/60.,
         LNUM.QUE,
         REFUSED/(REFUSED+NO.DONE),
         FDCNE,
         LNUM.QUE,
         LAVG.QUE/60.,
         MU.LWAIT/60.,
         LNUM,
         LQWAIT
         THUS
 **.**   *****     ****.**     ****.**L         ****    *.*******
   #XFERS    #WAIT IN QUE    AVG. WAIT IN QUE     AVG. WAIT
    ******     ******          *****.***           *****.***
    ******LNUM ****** LQWAIT

       GC NEXT
```

```
ELSE
   GC TC NCW2 OR ALT2  PER CASE
'NOW2'
      PRINT 2 LINE WITH
         UTILIZATION,
         NO.DONE,
         MEAN.WAIT/60.,
         AVG.QUE/60.,
         AVG.FULL/60.,
         NUM.QUE,
         INQUE.AVG,
         LNUM,
         LAVG/60.,
         QWAIT
         THUS
 **.**  *****  ****.**    *****.**  *****.**     ****    *****
                         *****L.REQS *****.**W/LXF ***** QWAIT

      SKIP 1 OUTPUT LINE
      PRINT 2 LINES WITH
         AVG.FULL/60.,
         MEAN.WAIT/60.,
         INQUE.AVG,
         AVG.QUE/60.,
         ANCT.EMP.Q,
         UTILIZATION
           THUS
SYS.T    QUE.T    N QUE   NE.QUE.T  NNE.Q   UTILIZ
***.**M  ***.**M  ***.**  ***.**M  ****.**  *.***
     GO CNEXT
'ALT2'
      PRINT 4 LINE WITH
         UTILIZATION,
         NC.DONE,
         AVG.QUE/60.,
         AVG.FULL/60.,
         IAFULL/60.,
         NUM.QUE,
         INQUE.AVG,
         EDCNE,
         LNUM.QUE,
```

```
          LAVG.QUE/60.,
          MU.LWAIT/60.,
          LQWAIT, QWAIT
          THUS
 **.**  *****  ****.**    *****.**   *****.**L     ****     *****
  #XFERS      #WAIT IN QUE    AVG. WAIT IN QUE     MU.LWAIT
  *******       ******          ****.***             ****.***
                ****** LQWAIT                      ***** QWAIT

      PRINT 3 LINES WITH
          AVG.FULL/60.,
          MEAN.WAIT/60.,
          INQUE.AVG,
          AVG.QUE/60.,
          ANCT.EMP.Q,
          UTILIZATION,
          IAFULL/60.
             THUS
SYS.T     QUE.T     N QUE   NE.QUE.T  NNE.Q   UTILIZA
***.**M   ***.**M   ***.**  ***.**M  ****.**  *.***
***.**M   TO THE CPU

      SKIP 1 OUTPUT LINE

      PRINT 1 LINE WITH
          XAFULL/60.,
          MU.LWAIT/60.,
          LINQUE.AVG,
          LAVG.QUE/60.
             THUS
***.**M   ***.**M   ***.**  ***.**M          FOR XFERS


     GO ANEXT

'CNEXT'

      PRINT 1 LINE THUS
               SYS.T     WAIT.T      QUEUE.T
      FOR I = 1 TO 14

      PRINT 1 LINE WITH
          ((I-1)*3)+1, I*3, FHISTO(I), WHISTO(I), QHISTO(I)
      THUS
 ** < T < **     **        **        **

PRINT 1 LINE WITH FHISTO(15), WHISTO(15), QHISTO(15) THUS
```

```
    T > 20      **        **        **

     GO NEXTR

'ANEXT'

   PRINT 1 LINE THUS
              TO CPU     TO TERM   WAIT.T   QUEUE.T

   FOR I = 1 TO 14
      PRINT 1 LINE WITH
        ((I-1)*3)+1, I*3, LFHISTO(I), FHISTO(I), WHISTO(I), QHISTO(I)
           THUS
 ** < T < **     **        **        **        **

   PRINT 1 LINE WITH LFHISTO(15), FHISTO(15), WHISTO(15), QHISTO(15)
      THUS
     T > 20     **        **        **        **

     GO NEXT
'NEXTR'
   SKIP 1 OUTPUT LINE
   PRINT 1 LINE THUS
                INTR/SESS       XMIT/SESS

   FOR I = 1 TO 11,
   PRINT 1 LINE WITH (I-1)/10 , I/10,  IRHISTO(I), TRHISTO(I)
      THUS
 *.* < X < *.*       ****            ****

'NEXT'
   PERFORM H.STATS

   RETURN

END
```

```
''*************************************************************
''                      HOST STATISTICS ROUTINE
''*************************************************************

ROUTINE FOR H.STATS



   PRINT 1 LINE THUS
HOST   UTILIZATION


   FOR EACH HOST,
      DO
      PRINT 1 LINE WITH
         HOST, HPLAIN.USE
         THUS
 **   ***.**
      LOOP

   RETURN

END
```

```
''*******************************************************************
''                              REPORT ROUTINE
''*******************************************************************

ROUTINE FOR REPORT
      DEFINE UTILS AS  VARIABLES
      DEFINE HRS, MIN, SEC AS INTEGER VARIABLES

      PRINT 2 LINES WITH
            TIME.V/60.,  NO.DONE
              THUS
  SIMULATION ENDED AT ****.*** MINUTES   DID ***** TASKS


   PRINT 1 LINE THUS
    PATH    UTILIZATION   AVG.QUEUE LENGTH
      FOR EACH TASHOST,
      DO
      LET UTILS = FLOW.CAPACITY(TASHOST) * 1.0
         PRINT 1 LINE WITH TASHOST, THUSE(TASHOST)/UTILS,
         AVG.TH.Q(TASHOST)   THUS
     **      **.**           ***.**

      LOOP
IF CASE = ALTERNATE,
      PRINT 3 LINES THUS
            HI-SPEED RESOURCE UTILIZATION

   PATH    UTILIZATION   AVG. QUEUE LENGTH

      FOR EACH LPATH,
         DO
            PRINT 1 LINE WITH
               LPATH,
               PATH.USE(LPATH)/1.0,
               AV.PATH.Q
               THUS
    **      **.**           ***.**
         LOOP
REGARDLESS
      RETURN

END
```

```
''*************************************************************
''                    AAGAIN ROUTINE
''*************************************************************

ROUTINE FOR AAGAIN


          FOR EACH TASHOST,
          FOR EACH TASK IN THQUEUE(TASHOST)
             DO
             REMOVE THE FIRST TASK FROM THQUEUE(TASHOST)
             DESTROY THE TASK
             LOOP

          FOR EACH TAS,
          FOR EACH UTASK IN UQUEUE(TAS)
             DO
             REMOVE THE FIRST UTASK FROM UQUEUE(TAS)
             DESTROY THE UTASK
             LOOP


          FOR EACH HOST,
          FOR EACH STASK IN SQUEUE(HOST)
             DO
             REMOVE THE FIRST STASK FROM SQUEUE(HOST)
             DESTROY THE STASK
             LOOP


          FOR EVERY TAS,
             DO
             LET TNUM(TAS) = 0
             LET TEATH(TAS) = 0
             LOOP
          FOR EVERY HOST,
             DO
             LET HNUM(HOST) = 0
             LET HEATH(HOST) = 0
             LOOP

          LET TIME.V = 0
          LET COMPLETED = 0
          LET REFUSED = 0
          LET USAGE = 0.
          LET TUSAGE = 0.
```

```
            LET  CWAIT = 0
            LET  NOWAIT = 0
            LET  WAIT.TIME =  0.0
            LET  LQWAIT = 0
            LET  XFULL.TIME  = 0
            LET  LINQUE = 0.
            LET  NOT.EMP.Q =  0.
            LET  INQUEUE = 0
            LET  DURATION =  0.0
            LET   LCOMPLETED  = 0
            RESET TOTALS OF WAIT.TIME,DURATION, FULL.TIME, QUE.WAIT.TIME
            RESET TOTALS OF USAGE, TUSAGE
            RESET TOTALS OF LFULL.TIME,XFULL.TIME
            RESET TOTALS OF LWAIT.TIME, LQUE.WAIT
            RESET TOTALS OF L1WAIT.TIME, L2WAIT.TIME, L1FULL.TIME,
               L2FULL.TIME, LRES.TIME,LQUE.WAIT, INQUEUE,
               LINQUE, NOT.EMP.Q
            FOR EACH TASHOST,
               DO
               RESET TOTALS OF THBUSY(TASHOST)
               RESET TOTALS OF N.THQUEUE(TASHOST)
               LOOP
            FOR EACH LPATH,
               DO
               RESET TOTALS OF BUSY(LPATH)
               RESET TOTALS OF N.LQUEUE(LPATH)
               LOOP
            FOR EACH HOST,
               DO
''             RESET TOTALS OF SWAIT.TIME(HOST), SFULL.TIME(HOST)
               RESET TOTALS OF N.HOST(HOST)
               LOOP

''          FOR EACH TAS,
''             DO
''             RESET TOTALS OF UWAIT.TIME(TAS), UFULL.TIME(TAS)
''             LOOP
            RETURN

END
```

```
''**************************************************************
''                         AGAIN ROUTINE
''**************************************************************

ROUTINE FCR AGAIN

      DEFINE NEXT AS AN INTEGER VARIABLE

      IF DATA IS ENDED, PRINT 2 LINES THUS

         NO MORE INPUT::::::: END OF RUN
         STOP
      ELSE
      READ NEXT
         IF NEXT = 999,
            FOR EACH TASHOST,
            FOR EACH TASK IN THQUEUE(TASHOST)
               DO
               REMOVE THE FIRST TASK FROM THQUEUE(TASHOST)
               DESTROY THE TASK
               LOOP
            FOR EVERY TAS,
               LET TNUM(TAS) = 0
            FOR EVERY HOST,
               LET HNUM(HOST) = 0
            LET TIME.V = 0
            LET COMPLETED = 0
            LET REFUSED = 0
            LET CWAIT = 0
            LET NOWAIT = 0
            RESET TOTALS OF WAIT.TIME,DURATION, FULL.TIME,

            FOR EACH TASHOST,
               DO
               RESET TOTALS OF THBUSY(TASHOST)
               RESET TOTALS OF N.THQUEUE(TASHOST)
               LOOP

            RETURN
         ELSE
         PRINT 1 LINE THUS
      DID NCT GET FLAG ::MORE:: TO CONTINUE, TERMINATING RUN
      STOP

END
```

```
''*****************************************************************
''                         TAS1.ARRIVAL  EVENT
''*****************************************************************

EVENT TAS1.ARRIVAL

      DEFINE TCHOICE, HCHOICE, TEMP AS INTEGER VARIAELES
      DEFINE DIRECT AS AN INTEGER VARIABLE


      LET TCHOICE = 1
      SCHEDULE A TAS1.ARRIVAL
''       IN EXPONENTIAL.F(1/LAMBDA(1),12+TCHOICE) * 60.0 SECONDS
   IN EXFCNENTIAL.F(1/LAMBDA(1), 12 + TCHOICE) * 3600.0 SECONDS
      PERFORM FINISH.UP GIVEN TCHOICE
      RETURN
END
```

```
''*************************************************************************
''                              TAS2.ARRIVAL  EVENT
''*************************************************************************

EVENT TAS2.ARRIVAL

      DEFINE TCHOICE, HCHOICE, TEMP AS INTEGER VARIABLES
      DEFINE DIRECT AS AN INTEGER VARIABLE


      LET TCHOICE = 2
      SCHEDULE A TAS2.ARRIVAL
          IN EXPONENTIAL.F(1/LAMBDA(2),12+TCHOICE) *3600.0 SECONDS
      PERFORM FINISH.UP GIVEN TCHOICE
      RETURN
END
```

```
''*************************************************************
''                       TAS3.ARRIVAL  EVENT
''*************************************************************

EVENT TAS3.ARRIVAL

     DEFINE TCHOICE, HCHOICE, TEMP AS INTEGER VARIABLES
     DEFINE DIRECT AS AN INTEGER VARIABLE


     LET TCHOICE = 3
     SCHEDULE A TAS3.ARRIVAL
        IN EXPONENTIAL.F(1/LAMBDA(3),12+TCHOICE) *3600.0 SECCNDS
     PERFORM FINISH.UP GIVEN TCHOICE
     RETURN
END
```

```
''**********************************************************************
''                        TAS4.ARRIVAL  EVENT
''**********************************************************************

EVENT TAS4.ARRIVAL

      DEFINE TCHOICE, HCHOICE, TEMP AS INTEGER VARIABLES
      DEFINE DIRECT AS AN INTEGER VARIABLE


      LET TCHOICE = 4
      SCHEDULE A TAS4.ARRIVAL
          IN EXPONENTIAL.F(1/LAMBDA(4),12+TCHOICE) *3600.0 SECONDS
      PERFORM FINISH.UP GIVEN TCHOICE
      RETURN
END
```

```
''*********************************************************************
''                          TAS5.ARRIVAL  EVENT
''*********************************************************************

EVENT TAS5.ARRIVAL

      DEFINE TCHOICE, HCHOICE, TEMP AS INTEGER VARIABLES
      DEFINE DIRECT AS AN INTEGER VARIABLE


      LET TCHOICE = 5
      SCHEDULE A TAS5.ARRIVAL
         IN EXPONENTIAL.F(1/LAMBDA(5),12+TCHOICE) * 3600.0 SECONDS
      PERFORM FINISH.UP GIVEN TCHOICE
      RETURN
END
```

```
''**********************************************************************
''                          TAS6.ARRIVAL  EVENT
''**********************************************************************

EVENT TAS6.ARRIVAL

      DEFINE TCHOICE, HCHOICE, TEMP AS INTEGER VARIABLES
      DEFINE DIRECT AS AN INTEGER VARIABLE


      LET TCHOICE = 6
      SCHEDULE A TAS6.ARRIVAL
         IN EXPONENTIAL.F(1/LAMBDA(6),12+TCHOICE) * 3600.0 SECONDS
      PERFORM FINISH.UP GIVEN TCHOICE
      RETURN
END
```

```
''*************************************************************
''                       TAS7.ARRIVAL  EVENT
''*************************************************************

EVENT TAS7.ARRIVAL

      DEFINE TCHOICE, HCHOICE, TEMP AS INTEGER VARIABLES
      DEFINE DIRECT AS AN INTEGER VARIABLE


      LET TCHOICE = 7
      SCHEDULE A TAS7.ARRIVAL
         IN EXPONENTIAL.F(1/LAMBDA(7),12+TCHOICE)  * 3600.0 SECONDS
      PERFORM FINISH.UP GIVEN TCHOICE
      RETURN
END
```

```
''***************************************************************
''                        TAS8.ARRIVAL  EVENT
''***************************************************************

EVENT TAS8.ARRIVAL

      DEFINE TCHOICE, HCHOICE, TEMP AS INTEGER VARIABLES
      DEFINE DIRECT AS AN INTEGER VARIABLE


      LET TCHOICE = 8
      SCHEDULE A TAS8.ARRIVAL
         IN EXPONENTIAL.F(1/LAMBDA(8),12+TCHOICE) * 3600.0 SECONDS
      PERFORM FINISH.UP GIVEN TCHOICE
      RETURN
END
```

```
''*************************************************************************
''                          FINISH.UP ROUTNE
''*************************************************************************

ROUTINE FOR FINISH.UP GIVEN TCHOICE

   DEFINE HCHOICE, TCHOICE, TEMP AS INTEGER VARIABLES
   DEFINE SCREENS, DATUS AS VARIABLES
   DEFINE DIRECT AS AN INTEGER VARIABLE


      PERFORM DB.SELECTION GIVEN TCHOICE YIELDING HCHOICE

   IF HCHCICE = 0,
         RETURN
   REGARDLESS
      PERFORM W.PROFILE GIVEN HCHOICE
            YIELDING SCREENS, DATUS
      LET TASHOST =  (TCHOICE -1) * N.HOST + HCHOICE



''-----------------------------------------------------------------------
''  MUST FIRST CHECK TO SEE IF THE REQUESTED PATH IS AVAILABLE
'' IF NOT AVAILABLE, WILL PUT SERVICE-REQUEST IN THE QUEUE
''  OTHERWISE WILL SET THE PATH BUSY AND SCHEDULE A TERMINATION
''  AT THE END OF SESSION.LENGTH
''-----------------------------------------------------------------------

      IF TNUM(TCHOICE) < TAS.MAX(TCHOICE)
         ANC
         HNUM(HCHOICE) < HOST.MAX(HCHOICE),

''-----------------------------------------------------------------------
''  SINCE PATH IS AVAILABLE, NOW MUST CHECK TO SEE WHETHER OR
''  NCT A SERVER-TAS IS INVOLVED IN THIS TRANSACTION.
''  IF A SERVER-TAS IS INVOLVED, MUST ADJUSTED THE APPRCPRIATE
''  ACCOUNTING COUNTERS TO REFLECT ITS USAGE, IN PARTICULAR
''  IF AS A ::TAS::, THEN MUST ALSO ADJUST THE ::HOST::
''    ACCCUNTING FLAG.
'' SIMILARLY, IF AS A ::HOST::, THEN MUST ADJUST THE ::TAS::
''     ACCOUNTING FLAG.
''-----------------------------------------------------------------------
            LET DURATION =  SCREENS
            LET LONG.FLAG =  DATUS
            LET DIRECT = 0
```

```
            PERFORM S.FLAG GIVEN TCHOICE, HCHOICE, DIRECT, TASHOST
            LET WAIT.TIME = 0.0
            ADD 1 TO NOWAIT
''----------------------------------------------------------------------
'' MUST DETERMINE WHETHER OR NOT THERE WILL BE SOME VOLUMINOUS OUTPUT
'' GENERATED FOR THIS INTERACTIVE SESSION.  IF SO, MUST MAKE
'' ARRANGEMENTS FOR ITS TRANSFER
''----------------------------------------------------------------------

         IF LONG.FLAG > 0,
               SCHEDULE A USEND GIVEN DURATION, TCHOICE, HCHOICE,
                  TASHOST, LONG.FLAG IN DURATION *60.0 SECONDS

               IF CASE = CURRENT,
                  LET X = (((LONG.FLAG * 11.)/RATE1) / 60.)
                  LET USAGE = DURATION/ ((((LONG.FLAG* 11.)/RATE1) /60.)
                     + DURATION)
                  LET TUSAGE = X / (X + DURATION)

               ELSE
                  LET USAGE = 1.
               ALWAYS
         ELSE
               LET FULL.TIME = DURATION * 60.
               LET LFULL.TIME = FULL.TIME
               SCHEDULE A THDEPART GIVEN  TASHOST, TCHOICE, HCHOICE,
                  DUMMY
                  IN DURATION * 60.00 SECONDS

               LET USAGE = 1.
         ALWAYS

      ELSE
''----------------------------------------------------------------------
'' IF NO QUEUE'S ARE PERMITTED, THE ARRIVAL WILL BE CANCELLED
''----------------------------------------------------------------------
         IF TURN.AWAY = 1,
            ADD 1 TO REFUSED
            RETURN
         ELSE

''----------------------------------------------------------------------
'' QUEUE'S ARE PERMITTED BUT
'' THE REQUESTED PATH IS BUSY, SO WILL PUT REQUEST IN THE QUEUE
''----------------------------------------------------------------------
               LET DURATION = SCREENS
               LET LONG.FLAG = DATUS
```

```
                CREATE A TASK
                LET ARRIVAL.TIME(TASK) = TIME.V
                LET TAS.CHOICE(TASK) = TCHOICE
                LET HOST.CHOICE(TASK) = HCHOICE
                LET INTER(TASK) = DURATION
                LET LLONG(TASK) = LONG.FLAG
                FILE TASK IN THQUEUE(TASHOST)
                ADD 1 TO INQUEUE
                LET NCT.EMP.Q = INQUEUE

        REGARDLESS

        RETURN

    END
```

```
''*************************************************************************
''                        TAS-HOST DEPART    EVENT
''*************************************************************************

EVENT THDEPART GIVEN  TASHOST, TAS,HOST, AMT.DATA

      DEFINE HSERVER, TUSER, TEMP  AS INTEGER VARIABLES
      DEFINE FIRST, DIRECT AS INTEGER VARIABLES
      DEFINE TASHOST, TAS, HOST AS INTEGER VARIABLES
      DEFINE TTASHOST, HHSERVER, TTUSER AS INTEGER VARIABLES
      DEFINE  THE.TASK, THE.QUE AS INTEGER VARIABLES
      DEFINE THIS.ONE AS AN INTEGER VARIABLE
      DEFINE EARLY.TIME, AMT.DATA AS VARIABLES
      DEFINE SEE AS AN INTEGER VARIABLE

''-----------------------------------------------------------------------
'' FIRST CRDER OF BUSINESS IS TO PROCESS THE DEPARTURE THAT
'' CAUSED ENTRY INTO THIS PROCESS.
'' BY PRCCESS: MEAN SIMPLY THE UPDATING OF THE SYSTEM STATUS
''
'' AFTER THAT IS DONE, WE MUST LOOK FOR OTHER JOBS THAT ARE
'' COMPLETED AT THIS TIME INSTANT AND MAKE THE APPROPRIATE
'' SYSTEM ADJUSTMENTS BEFORE PROCEEDING TO ASSIGNING ANY
'' NEW WORK.
''-----------------------------------------------------------------------

   ADD 1 TC LCOMPLETED
   LET DIRECT = 1
   PERFORM S.FLAG GIVEN TAS, HOST, DIRECT, TASHOST

   FOR EACH THDEPART IN EV.S.(I.THDEPART)
      DO
         LET SEE = (TIME.A(THDEPART) - TIME.V) * 100000.00

         IF SEE = 0 ,
''-----------------------------------------------------------------------
'' HAVE FCUND ANOTHER JOB TO RELEASE AT THIS SAME TIME INSTANT.
'' SO WILL TAKE STEPS TO REMOVE THIS JOB FROM THE DEPARTURE-QUEUE
'' AND UPDATE THE SYSTEM STATUS, BEFORE PROGRESSING TO
'' INITIATING ANY NEW WORK.
''-----------------------------------------------------------------------

            LET TTASHOST =TTOHOST(THDEPART)
            ADD 1 TO LCOMPLETED
            LET DIRECT = 1
            LET TAS = TTAS(THDEPART)
```

```
            LET HOST = HHOST(THDEPART)
            PERFORM S.FLAG GIVEN TAS, HOST, DIRECT, TTASHOST
            CANCEL THIS THDEPART

        REGARDLESS
    LOOP

''------------------------------------------------------------------
''  SECOND ORDER OF BUSINESS IS TO LOOK OVER THOSE JOB-REQUESTS
''   IN THE QUEUES
''  TWO CONDITIONS MUST BE MET BEFORE A REQUEST WILL BE TAKEN
''  OFF THE QUEUE.
''   FIRST WE LOOK AT ALL THE CANDIDATE QUEUES, I.E.
''  THAT MEET THE CRITERIA THAT AN APPROPRIATE PATH IS AVAILABLE
''   SECOND, OF ALL THE APPROPRIATLY AVAILABLE QUEUES,
''  WE MUST FIND THE JOB-REQUEST WITH THE EARLIEST TIME.
''------------------------------------------------------------------


'RELOOK'
  LET THE.QUE = 0
  PERFORM QLOOK YIELDING THE.QUE

''    WANT TO MAKE CERTAIN THAT A CANDIDATE HAS BEEN FOUND
      IF THE.QUE = 0, GO NONE
      ELSE
         LET TASHOST = THE.QUE
         REMOVE THE FIRST TASK FROM THQUEUE(TASHOST)
         SUBTRACT 1 FROM INQUEUE
         IF INQUEUE NE 0, LET NOT.EMP.Q = INQUEUE
         ALWAYS
         LET WAIT.TIME = TIME.V - ARRIVAL.TIME(TASK)
         LET QUE.WAIT.TIME = WAIT.TIME
         ADD 1 TO QWAIT

         LET TUSER = TAS.CHOICE(TASK)
         LET HSERVER = HOST.CHOICE(TASK)
''------------------------------------------------------------------
''  CHECK WHETHER OR NOT A SERVER-TAS IS INVOLVED IN THIS WORK-REQUEST
''  IF YES, THEN WE MAKE THE APPROPRIATE ADJUSTMENTS IN THE
''  ACCOUNTING INFORMATION
''------------------------------------------------------------------

         LET DIRECT = 0
         PERFORM S.FLAG GIVEN TUSER, HSERVER, DIRECT, TASHOST
```

```
IF LLONG(TASK) > 0.,
      SCHEDULE A USEND GIVEN (WAIT.TIME/60.) + INTER(TASK),
         TUSER, HSERVER, TASHOST, LLONG(TASK)
         IN INTER(TASK) * 60 SECONDS
      IF CASE = CURRENT,
         LET X = (((LONG.FLAG * 11.)/RATE1) / 60.)
         LET USAGE= INTER(TASK)/((((LLONG(TASK)*11.)/RATE1)/ 60.)
         + INTER(TASK))
         LET TUSAGE = X/(X+DURATION)

      ELSE
         LET USAGE = 1.
      ALWAYS
ELSE
      LET FULL.TIME = WAIT.TIME + (INTER(TASK) * 60.)
      LET LFULL.TIME = FULL.TIME
      SCHEDULE A THDEPART GIVEN TASHOST, TAS.CHOICE(TASK),
         HOST.CHOICE(TASK), DUMMY IN INTER(TASK) * 60.0 SECONDS
      LET USAGE = 1.
ALWAYS
DESTROY TASK
''---------------------------------------------------------------------
''    HERE WANT TO CHECK WHETHER OR NOT SHOULD REMOVE
''    ANCTHER TASK.  THIS IS DEPENDENT ON WHETHER OR NOT
''    MAX IS MET.
''---------------------------------------------------------------------
      GO RELOOK
'NONE'

      RETURN

END
```

```
''*************************************************************
''                         S-FLAG ROUTINE
''*************************************************************

ROUTINE FOR S.FLAG GIVEN TTT,HHH,DIRECTION, THE.WAY

   DEFINE TTT,HHH,TEMP, DIRECTION, THE.WAY AS INTEGER VARIABLES

   IF S1.FLAG(TTT) > 0,
      LET TEMP = HOSTNUM + S1.FLAG(TTT)
      IF DIRECTION = 0,
         ADD 1 TO HNUM(TEMP)
      ELSE
         SUBTRACT 1 FROM HNUM(TEMP)
      ALWAYS
   REGARDLESS

   IF S2.FLAG(HHH) > 0,
      LET TEMP = TASNUM + S2.FLAG(HHH)
      IF DIRECTION = 0,
         ADD 1 TO TNUM(TEMP)
      ELSE
         SUBTRACT 1 FROM TNUM(TEMP)
      ALWAYS
   REGARDLESS
   IF DIRECTION = 0,

      ADD 1 TO THBUSY(THE.WAY)
      ADD 1 TO WORK(THE.WAY)
      ADD 1 TO TNUM(TTT)
      ADD 1 TO HNUM(HHH)

   ELSE

      SUBTRACT 1 FROM THBUSY(THE.WAY)
      SUBTRACT 1 FROM WORK(THE.WAY)
      SUBTRACT 1 FROM TNUM(TTT)
      SUBTRACT 1 FROM HNUM(HHH)
   REGARDLESS
   RETURN
END
```

```
''*****************************************************************
''                          THQUEUE LOOP-UP ROUTINE
''*****************************************************************

ROUTINE FOR QLOOK YIELDING FOUND

DEFINE FCUND AS AN INTEGER VARIABLE
DEFINE EARLY.TIME AS A VARIABLE


      LET FCUND  = 0

      LET EARLY.TIME = RINF.C
      FOR EACH TASHOST  WITH
         WORK(TASHOST) < FLOW.CAPACITY(TASHOST)
         AND THQUEUE(TASHOST) NOT EMPTY   ,
         DO
            IF ARRIVAL.TIME(F.THQUEUE(TASHOST)) < EARLY.TIME,
               LET EARLY.TIME = ARRIVAL.TIME(F.THQUEUE(TASHOST))
               LET FCUND = TASHOST
            REGARDLESS
         LOOP
    RETURN

END
```

```
''*************************************************************
''                          USEND EVENT
''*************************************************************

EVENT USEND GIVEN XCHANG.TIME, T,H,TH,XFER.AMT

'' USER IS READY TO REQUEST DATA XFER.  CUSTOMER IS ATTACHED TO A T
'' HANDLE AMOUNT OF TIME REQUERED TO XMIT THE REQUEST FOR XFER TO T

DEFINE XCHANGE.TIME, XFER.AMT, SND.CMD.TIME, RES1.TIME AS VARIABLES
DEFINE T, H, TH AS INTEGER VARIABLES


GO TO SOLIS OR USIS OR NUIS  PER H

'SOLIS'
   LET SND.CMD.TIME = (UNIFORM.F(80.,100.,9+H) * 11.)/2400.
   GO NEXT

'USIS'
   LET SND.CMD.TIME = (UNIFORM.F (25.,50.,9+H) * 11.0)/2400.
   GO NEXT

'NUIS'
   LET SND.CMD.TIME = (UNIFORM.F (25.,50.,9+H) * 11.0)/2400.
   GO NEXT


'NEXT'
   LET RES1.TIME = SND.CMD.TIME + (XCHANG.TIME * 60.)
   SCHEDULE A UC.ARRIVAL GIVEN TH, T, H, XFER.AMT, SND.CMD.TIME,
      RES1.TIME
      IN SND.CMD.TIME SECONDS

   RETURN
END
```

```
''**********************************************************************
''                        UC.ARRIVAL EVENT
''**********************************************************************

EVENT UC.ARRIVAL GIVEN TH, T, H, X.AMT, C.TIME, R1.TIME
'' HANDLES CNE AT A TIME THE PROCESSING OF USER REQUEST FOR XFER

DEFINE TH, T, H AS INTEGER VARIABLES
DEFINE X.AMT, TU, C.TIME, R1.TIME AS VARIABLES


LET TU = .25

IF UBUSY(T) = OCCUPIED,
   CREATE A UTASK
   LET UARR.TIME(UTASK) = TIME.V
   LET TTU(UTASK) = TU
   LET UTTHH(UTASK) = TH
   LET UC.TAS(UTASK) = T
   LET UC.CTIME(UTASK) = C.TIME
   LET UC.HOST(UTASK) = H
   LET UC.AMT(UTASK) = X.AMT
   LET UC.RES(UTASK) = R1.TIME
   FILE UTASK IN UQUEUE(T)

ELSE
   LET UWAIT.TIME(T) = 0.0
   LET UFULL.TIME(T) = TU
   LET L1WAIT.TIME = UWAIT.TIME(T)
   LET L1FULL.TIME = UFULL.TIME(T)
   LET UBUSY(T) = OCCUPIED
   LET R1.TIME = R1.TIME + TU
   SCHEDULE A UC.DEPART GIVEN  TH, T, H, X.AMT, C.TIME, R1.TIME
      IN TU SECONDS

ALWAYS

   RETURN

END
```

```
''*****************************************************************
''                          UC.DEPART EVENT
''*****************************************************************

EVENT UC.DEPART GIVEN TASHOST, TAS, HOST, UX.AMT, UC.TIME, R2.TIME

''COMPLETED PROCESSING OF ONE USER REQUEST FOR XFER.
''   NEED TO COMPUTE AMT OF TIME REQUIRED TO XFER REQUEST TO SERVER
''   SCHEDULE ARRIVAL OF REQUEST AT SERVER HOST
''   CHECK FOR WORK IN QUEUE
''        NOT EMPTY:  DO WORK TO PROCESS NEXT REQUEST
''        EMPTY:  RETURN


DEFINE TASHOST, TAS, HOST AS INTEGER VARIABLES
DEFINE UX.AMT, FWD.CMD.TIME, UC.TIME, R2.TIME AS VARIABLES

LET FWD.CMD.TIME = UC.TIME/4.
LET R2.TIME = R2.TIME + FWD.CMD.TIME
LET UBUSY(TAS) = FREE
SCHEDULE A SC.ARRIVAL GIVEN TASHOST, TAS, HOST, UX.AMT,R2.TIME
   IN FWD.CMD.TIME SECONDS
IF UQUEUE(TAS) NOT EMPTY,
   REMOVE FIRST UTASK FROM UQUEUE
   LET TASHOST = UTTHH(UTASK)
   LET TAS = UC.TAS(UTASK)
   LET UWAIT.TIME(TAS) = TIME.V - UARR.TIME(UTASK)
   LET UFULL.TIME(TAS) = UWAIT.TIME(TAS) + TTU(UTASK)
   LET L1WAIT.TIME = UWAIT.TIME(TAS)
   LET L1FULL.TIME = UFULL.TIME(TAS)
   LET R2.TIME = UC.RES(UTASK) + UFULL.TIME(TAS)
   SCHEDULE A UC.DEPART GIVEN UTTHH(UTASK),UC.TAS(UTASK),
      UC.HOST(UTASK), UC.AMT(UTASK), R2.TIME
         IN TTU(UTASK) SECONDS
   LET UBUSY(TAS) = BUSY
   DESTROY UTASK

ELSE
   LET UBUSY(TAS) = FREE
ALWAYS
   RETURN

END
```

```
''**********************************************************************
''                               SC.ARRIVAL  EVENT
''**********************************************************************

EVENT SC.ARRIVAL GIVEN STH, ST, SH, SXMITT.AMT, R3.TIME

''HANDLES ONLY ONE REQUEST AT A TIME.  THIS IS SERVER HOST RECEIV
''REQUEST FOR XFER

DEFINE STH, ST, SH AS INTEGER VARIABLES
DEFINE SXMITT.TIME, R3.TIME,TS AS VARIABLES


LET TS = .5
IF SBUSY(SH) = OCCUPIED,
   CREATE STASK
   LET SARR.TIME(STASK) = TIME.V
   LET TTS(STASK) = TS
   LET TTHH(STASK) = STH
   LET TTT(STASK) = ST
   LET TTH(STASK) = SH
   LET SC.AMT(STASK) = SXMIT.AMT
   LET SC.RES(STASK) = R3.TIME
   FILE STASK IN SQUEUE(SH)

ELSE

   LET SWAIT.TIME(SH) = 0.0
   LET SFULL.TIME(SH) = TS
   LET SBUSY(SH) = OCCUPIED
   LET L2WAIT.TIME= SWAIT.TIME(SH)
   LET L2FULL.TIME = SFULL.TIME(SH)
   LET R3.TIME = R3.TIME + TS
   SCHEDULE A SC.DEPART GIVEN STH, ST, SH, SXMITT.AMT, R3.TIME
      IN TS SECONDS

ALWAYS

   RETURN

END
```

```
''***************************************************************************
''                              SC.DEPART EVENT
''***************************************************************************

EVENT SC.DEPART GIVEN TASHOST, TAS, HOST, XMIT.AMT, R4.TIME

''   SERVER HOST HAS COMPLETED PREPARING DATA FOR XFER.
''   PASS DATA FROM SERVER TO USER CPU

DEFINE ANSR1.TIME, XMIT.AMT, R4.TIME  AS VARIABLES
DEFINE TASHOST, TAS, HOST, DIRECT, THE.QUE AS INTEGER VARIABLES


LET SBUSY(HOST)  = FREE
IF CASE = CURRENT,
   LET ANSR1.TIME = (XMIT.AMT * 11.)/RATE1
   LET R4.TIME = R4.TIME + ANSR1.TIME
   SCHEDULE A LTHDEPART GIVEN TASHOST, TAS, HOST, XMIT.AMT, R4.TIME
      IN ANSR1.TIME SECONDS

ELSE
   IF TSPD.FLAG(TAS) = YES   AND   HSPD.FLAG(HOST) = YES,
         LET ANSR1.TIME = (XMIT.AMT * 11.)/RATE2
         LET DIRECT = 1
         PERFORM S.FLAG GIVEN TAS, HOST, DIRECT, TASHOST
         IF TPATH(TAS) = 0 AND HPATH(HOST) = 0,
            ADD 1 TO TPATH(TAS)
            ADD 1 TO HPATH(HOST)
            ADD 1 TO BUSY(TASHOST)
            LET LWAIT.TIME = 0.
            LET R4.TIME = R4.TIME + ANSR1.TIME
            SCHEDULE A LTHDEPART GIVEN TASHOST, TAS, HOST,
                  XMIT.AMT, R4.TIME
                  IN ANSR1.TIME SECONDS

         ELSE
            CREATE A LTASK
            LET LARR.TIME(LTASK) = TIME.V
            LET TAKER(LTASK) = TAS
            LET GIVER(LTASK) = HOST
            LET LENGTH(LTASK) = XMIT.AMT
            LET RES.TIME(LTASK) = R4.TIME
            FILE LTASK IN LQUEUE(TASHOST)
            ADD 1 TO LINQUE

         ALWAYS
```

```
    ELSE

          LET ANSR1.TIME = (XMIT.AMT * 11.)/RATE1
          LET R4.TIME = R4.TIME + ANSR1.TIME
          SCHEDULE A LTHDEPART GIVEN TASHOST, TAS, HOST,
                XMIT.AMT, R4.TIME
                IN ANSR1.TIME SECONDS

    ALWAYS
ALWAYS
'SEEAGAIN'
    LET THE.QUE = 0
    PERFORM CLOOK YIELDING THE.QUE
    IF THE.QUE = 0,
          GO SEENONE
    ELSE
          LET TASHOST = THE.QUE
          REMOVE THE FIRST TASK FROM THQUEUE(TASHOST)
          LET HOST = HOST.CHOICE(TASK)
          LET TAS = TAS.CHOICE(TASK)
          SUBTRACT 1 FROM INQUEUE
          IF INQUEUE NE 0, LET NOT.EMP.Q = INQUEUE
          ALWAYS
          LET WAIT.TIME = TIME.V - ARRIVAL.TIME(TASK)
          LET QUE.WAIT.TIME = WAIT.TIME
          ADD 1 TO QWAIT
          LET DIRECT = 0
          PERFORM S.FLAG GIVEN TAS, HOST, DIRECT, TASHOST

          IF LLONG(TASK) > 0,
                SCHEDULE A USEND GIVEN INTER(TASK) +(WAIT.TIME/60.),
                   TAS, HOST,
                   TASHOST, LLONG(TASK)
                   IN INTER(TASK) * 60. SECONDS
                IF CASE = CURRENT,
                   LET USAGE= INTER(TASK)/((((LLONG(TASK)*11.)/RATE1)/ 60.)
                         + INTER(TASK))
                ELSE
                   LET USAGE = 1.
                ALWAYS

          ELSE
                LET FULL.TIME = (INTER(TASK) * 60.) + WAIT.TIME
                LET LFULL.TIME = FULL.TIME
                SCHEDULE A THDEPART GIVEN TASHOST, TAS, HOST, DUMMY
                   IN INTER(TASK) * 60. SECONDS
                LET USAGE = 1.
```

```
        ALWAYS
   DESTROY TASK
   GO SEEAGAIN

'SEENONE'

IF SQUEUE(HOST) NOT EMPTY,
   REMOVE FIRST STASK FROM SQUEUE(HOST)
   LET HOST = TTH(STASK)
   LET TAS = TTT(STASK)
   LET TASHOST = TTHH(STASK)
   LET SWAIT.TIME(HOST) = TIME.V - SARR.TIME(STASK)
   LET SFULL.TIME(HOST) = SWAIT.TIME(HOST) + TTS(STASK)
   LET R4.TIME = SC.RES(STASK) + SFULL.TIME(HOST)
   LET SBUSY(HOST) = OCCUPIED
   SCHEDULE A SC.DEPART GIVEN TASHOST, TAS, HOST, SC.AMT(STASK),
      R4.TIME
      IN TTS(STASK) SECONDS
   DESTROY STASK

ELSE

   LET SBUSY(HOST) = FREE

ALWAYS

   RETURN

END
```

```
''**************************************************************
''                       LTHDEPART  EVENT
''**************************************************************

EVENT LTHDEPART GIVEN TASHOST, TAS, HOST, X.AMT, R5.TIME

''   DATA HAS ARRIVED AT USER CPU AND MUST BE FORWARDED TO TERMINAL
''   ONLY THOSE REQUIRING LONG XFER WILL COME THROUGH THIS EVENT

DEFINE TASHOST, TAS, HOST, SEE, DIRECT, THE.QUE AS INTEGER VARIABLES
DEFINE X.AMT, TO.USER.TIME, R5.TIME, EARLY.TIME  AS VARIABLES


PERFORM LSTAT GIVEN TAS, HOST, TASHOST, X.AMT, R5.TIME
   YIELDING TO.USER.TIME, R5.TIME

IF CASE = ALTERNATE AND TSPD.FLAG(TAS) = YES AND HSPD.FLAG(HOST) = YES,
   PERFORM CASE2 GIVEN TASHOST, TAS, HOST
ELSE
   SCHEDULE A THDEPART GIVEN TASHOST, TAS, HOST, R5.TIME
''      IN TO.USER.TIME SECONDS
      NOW

ALWAYS


'EVNT.CHECK'
FOR EACH LTHDEPART IN EV.S.(I.LTHDEPART)
   DO
      LET SEE =  (TIME.A(LTHDEPART) - TIME.V) * 100000.
      IF SEE = 0,
         LET TASHOST = WWAY(LTHDEPART)
         LET TAS = WTAS(LTHDEPART)
         LET HOST = WHOST(LTHDEPART)
         LET X.AMT = WDATA.AMT(LTHDEPART)
         LET R5.TIME = WRES.TIME(LTHDEPART)
         PERFORM LSTAT GIVEN TAS, HOST, TASHOST, X.AMT, R5.TIME
            YIELDING TO.USER.TIME, R5.TIME

         IF CASE = ALTERNATE
               AND TSPD.FLAG(TAS) = YES AND HSPD.FLAG(HOST) = YES,
                     PERFORM CASE2 GIVEN TASHOST, TAS, HOST
         ELSE
               SCHEDULE A THDEPART GIVEN TASHOST, TAS,HOST, R5.TIME
''                IN TO.USER.TIME SECONDS
               NOW
```

```
          REGARDLESS

          CANCEL THIS LTHDEPART
       REGARDLESS
    LOOP
''------------------------------------------------------------------
''   HAVING FREE'ED UP RESOURCES, NOW WANT TO GO THROUGH QUEUE FOR
''   HI-SPEED XFERS AND START THE ONE WITH THE EARLIEST TIME
''------------------------------------------------------------------

'RELOOK'
    LET THE.QUE = 0
    LET EARLY.TIME = RINF.C

    IF CASE = ALTERNATE,

       FOR EACH TASHOST WITH
          BUSY(TASHOST) = FREE
          AND LQUEUE(TASHOST) NOT EMPTY,
          DO
             IF LARR.TIME(F.LQUEUE(TASHOST)) < EARLY.TIME,
                LET EARLY.TIME = LARR.TIME(F.LQUEUE(TASHOST) )
                LET THE.TAS = TAKER(F.LQUEUE(TASHOST))
                LET THE.HOST = GIVER(F.LQUEUE(TASHOST))
                LET THE.QUE = TASHOST
             REGARDLESS
          LOOP

       IF THE.QUE = 0, GO NONE
       ELSE
          LET TASHOST = THE.QUE
          LET TAS = THE.TAS
          LET HOST = THE.HOST
          ADD 1 TO TPATH(TAS)
          ADD 1 TO HPATH(HOST)
          ADD 1 TO BUSY(TASHOST)
          PERFORM LCONTINUE GIVEN TASHOST
          GO RELOOK
    REGARDLESS
'NONE'
    RETURN

END
```

```
''*****************************************************************
''                          LSTAT ROUTINE
''*****************************************************************

ROUTINE FOR LSTAT GIVEN TAS,HOST,ROUTE, AMT,LSCUM.TIME
      YIEIDING PAUSE,R6.TIME

DEFINE ROUTE,TAS, HOST AS  INTEGER VARIABLES
DEFINE AMT, PAUSE, R6.TIME, LSCUM.TIME AS VARIABLES

'' ADD 1 TC LCOMPLETED
LET PAUSE = (AMT * 11.)/2400.
LET R6.TIME= LSCUM.TIME + PAUSE
''IF CASE = ALTERNATE AND TSPD.FLAG(TAS) = YES AND HSPD.FLAG(HOST) = YES,
IF CASE = ALTERNATE,
      LET FULL.TIME = R6.TIME
      LET LFULL.TIME =LSCUM.TIME
      ADD 1 TO LCOMPLETED
ELSE
      LET FULL.TIME = LSCUM.TIME
      IET LFULL.TIME = FULL.TIME
ALWAYS
      LET LRES.TIME = LSCUM.TIME

RETURN

END
```

```
''*************************************************************
''                           LCONTINUE ROUTINE
''*************************************************************
ROUTINE FCR LCONTINUE GIVEN TASHOST

DEFINE R8.TIME,ANSR1.TIME AS VARIABLES
DEFINE TASHCST AS INTEGER VARIABLE

REMOVE THE FIRST LTASK FROM LQUEUE(TASHOST)
LET LWAIT.TIME = TIME.V - LARR.TIME(LTASK)
LET LQUE.WAIT = LWAIT.TIME
ADD 1 TC LQWAIT
SUBTRACT 1 FROM LINQUE
LET ANSR1.TIME = (LENGTH(LTASK) * 11.)/RATE2
LET R8.TIME = RES.TIME(LTASK) + LWAIT.TIME + ANSR1.TIME
SCHEDULE A LTHDEPART GIVEN TASHOST, TAKER(LTASK), GIVER(LTASK),
   LENGTH(LTASK), R8.TIME
   IN ANSR1.TIME SECCNDS
DESTRCY LTASK

RETURN

END
```

```
''**********************************************************
''                                   CASE2 ROUTINE
''**********************************************************

ROUTINE FOR CASE2 GIVEN TASHOST, TAS, HOST

''   LOOKING FOR TO SEE IF THERE IS ANOTHER LONG XFER TO DO ON THIS
''   SAME RESOURCE-PATH

DEFINE TASHOST,TAS,HOST AS  INTEGER VARIABLES

IF LQUEUE(TASHOST) NOT EMPTY,
   PERFORM LCONTINUE GIVEN TASHOST
ELSE
   SUBTRACT 1 FROM TPATH(TAS)
   SUBTRACT 1 FROM HPATH(HOST)
   SUBTRACT 1 FROM BUSY(TASHOST)

ALWAYS

   RETURN

END
```

```
''***************************************************************
''                        CLOSING  EVENT
''****************************************************************

EVENT CLCSING

      CANCEL THE TAS1.ARRIVAL
      CANCEL THE TAS2.ARRIVAL
      CANCEL THE TAS3.ARRIVAL
      CANCEL THE TAS4.ARRIVAL
''    CANCEL THE TAS5.ARRIVAL
''    CANCEL THE TAS6.ARRIVAL
''    CANCEL THE TAS7.ARRIVAL
''    CANCEL THE TAS8.ARRIVAL

   RETURN

END
```

```
''*******************************************************************
''                          DATA BASE SELECTION ROUTINE
''*******************************************************************

ROUTINE FOR DB.SELECT GIVEN WHO  YIELDING HCHOICE

DEFINE WHC, HCHOICE            AS INTEGER VARIABLE
DEFINE NET                     AS REAL VARIABLE


''------------------------------------------------------------------
''   DETERMINE WHETHER OR NOT REQUEST NETWORK SERVICE
''------------------------------------------------------------------
LET NET = RANDOM.F(20+WHO)
IF NET <= TSTUFF(WHO) ,

      LET WHICH = RANDOM.F(28+WHO)
      IF WHICH <= T.HOST(WHO,1),
         LET HCHOICE = 1
         GO NEXT
      REGARDLESS

      IF T.HOST(WHO,1) < WHICH  AND WHICH <= T.HOST(WHO,2),
         LET HCHOICE = 2
         GO NEXT
      REGARDLESS

      IF T.HOST(WHO,2) < WHICH AND WHICH <= T.HOST(WHO,3),
         LET HCHOICE = 3
         GO NEXT

      REGARDLESS
ELSE
      LET HCHOICE = 0
REGARDLESS

'NEXT'
       RETURN
END
```

```
''*************************************************************************
''                          W.PROFILE ROUTINE
''*************************************************************************

ROUTINE FOR W.PROFILE GIVEN HCHOICE YIELDING
      DURATION, LONG.FLAG


DEFINE W.SLOT, DURATICN, LONG.FLAG    AS VARIABLES
DEFINE HCHOICE   AS INTEGER VARIABLES

IF RANDOM.F(HCHOICE) <= MATRIX(HCHOICE,1),
''------------------------------------------------------------------
'' (ONLY INTER WORK)    COMPUTE LENGTH OF THE INTERACTIVE PORTION
''------------------------------------------------------------------
'' LET DURATION = EXPONENTIAL.F(1./MATRIX(HCHOICE,3),3+HCHOICE)
   GO TO UNIF1 OR NORM1 OR EXPO1 PER MATRIX(HCHOICE,2)

'UNIF1'
    LET DURATION = UNIFORM.F(MATRIX(HCHOICE,3),MATRIX(HCHOICE,4),
        3+HCHOICE)
    GC FINELSE
'NORM1'
    LET DURATION = NCRMAL.F(MATRIX(HCHOICE,3),MATRIX(HCHOICE,4),
        3+HCHOICE)
    GC FINELSE
'EXPO1'
   LET DURATION = EXPONENTIAL.F( MATRIX(HCHOICE,3),3+HCHOICE)
   GO FINELSE
'FINELSE'
   LET LCNG.FLAG = 0.0
ELSE

''------------------------------------------------------------------
''(INTER AND PRINT REQUEST) COMPUTE INTER AND DATA XFER REQUESTED
''------------------------------------------------------------------
'' LET DURATION = EXPONENTIAL.F(1./MATRIX(HCHOICE,9), 3+HCHOICE)
   GO TC UNIF2 OR NORM2 OR EXPO2 PER MATRIX(HCHOICE,8)

'UNIF2'
    LET DURATION =UNIFORM.F(MATRIX(HCHOICE,9),MATRIX(HCHOICE,10),
          3+HCHOICE)
    GO FINE2
'NORM2'
    LET DURATION = NCRMAL.F(MATRIX(HCHOICE,3),MATRIX(HCHOICE,4),
          3+HCHOICE)
    GO FINE2
```

```
'EXPO2'
   LET DURATION  = EXPONENTIAL.F(MATRIX(HCHOICE,9) ,  3+HCHOICE)
   GO FINE2
'FINE2'
   GO TO UNIF3 OR NORM3 OR EXPO3 PER MATRIX(HCHOICE,5)

'UNIF3'
    LET DURATION =UNIFORM.F (MATRIX(HCHOICE,6) ,MATRIX (HCHOICE,7) ,
          6+HCHOICE)
    GO FINE2
'NORM3'
    LET DURATION = NORMAL.F (MATRIX(HCHOICE,6) ,MATRIX (HCHOICE,7) ,
          6+HCHOICE)
    GO FINE2
'EXPO2'
   LET LONG.FLAG = EXPONENTIAL.F(    MATRIX(HCHOICE,6) ,HCHOICE+6)
   GO FINE3

'FINE3'
ALWAYS
RETURN
END
/*
//GO.SYSIN DD *
data cards
/*
//
```

LIST OF REFERENCES

1. Licklider, J.C.R., Vezza, A., "Applications of Information Networks", Proceedings of the IEEE, Vol. 66, No. 11, November 1978, pp. 1330-1346.

2. Roberts, L.G., Wessler, B.D., "Computer Network Development to Achieve Resource Sharing", AFIPS Conf. Proc., 1970 Spring Joint Computer Conference, Vol. 36, No. 19, pp. 543-549.

3. COINS Project Management Office, COINS Network Workload and Performance to 1990, by H. Kinslow Associates, Inc., December 1980.

4. COINS Project Management Office, Technical Proposal for Development of COINS Data Correlation Experiment, May 1982.

5. Defense Advanced Research Projects Agency (DARPA) Report P-82-1002-tp, Technical Proposal for: Secure Videodisc-Based Intelligence Information Display, by Interactive Television Co., February 1982.

6. Informal correspondence from Mr. George M. Hicken, COINS Project Manager, July 1982.

7. Informal correspondence from Dr. R.L. Wigington, Director, Research and Development, Chemical Abstracts Service, March, 1983.

8. Dominick, W.D., Penneman, W.D., "Monitoring and Evaluation of On-Line Information System Usage", Information Processing & Management, Vol. 16, No. 1., 1980, pp. 17-35.

9. Melnyk, V.," Man-machine Interface: Frustration", JASIS, November/December, 1972, pp. 392-401.

10. National Library of Medicine, Report No. NLM 78-7, Evaluation of the On-Line Process, by McDonald, D., Wanger, J., Cuadra Associates, Santa Monica, Ca., January 1980.

11. Baker, C.A., Eason, K.D., "An Observational Study of Man-Computer Interaction Using an Online Bibliographic Information Retrieval system", Online Review, Vol. 5, No. 2, April 1981, pp.121-132.

12. Benefeld, A.R., Kugel, R., Marcus, R.S., "Catalog Information and Text as Indicators of Relevance", JASIS, January 1978, pp. 16-30.

13. Carlisle, J., Martin, T.H., Treu, S., "The User Interface for Interactive Bibliographic Searching: An Analysis of the Attitudes of Nineteen Information Scientists", JASIS, March/April, 1973, pp. 142-147.

14. Marcus, R.S.," User Assistance in Bibliographic Retrieval Networks Through a Computer Intermediary", IEEE Transactions on Systems, Man, and Cybernetics, Vol. Smc-12, No. 2, March/April 1982.

15. Dayton, D.L., Lundeen, J.W., Pollock, J.J., "Automated Techniques for Online Search Guidance: A Review", 4th International Online Information Meeting, London, 9-11 December 1980, pp. 317-333.

16. Tedd, l. "Intelligence in the User's Terminal: A Look at Current Options and Possibilities", 5th International Online Information Meeting, London, 8-10 December 1981, pp. 1-10.

17. Laboratory for Information and Decision Systems, M.I.T., Cambridge, Ma., Rep. LIDS-R-1233, Investigations of Computer-Aided Document Search Strategies, Marcus, R.S., September 1, 1982.

18. Meadow, C.T., Epstein, B.E., "Individualized Instruction for Data Access", 1st International Online Information Meeting, London, 13-15 December 1977, pp. 179-194.

19. Durkin, K., Egeland, J., Garson, L., Terrant, S., "An Experiment to Study the Online Use of a Full-Text Primary Journal Database", 4th International Online Information Meeting, London, 9-11 December 1980, pp. 53-56.

20. Kiviat, P.J., Markowitz, H.M, Villanueva, R., SIMSCRIPT II.5 Programming Language, ed. Russel, E.C., C.A.C.I., Los Angeles, Ca., October, 1975.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Superintendent ATTN: Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Superintendent ATTN: Prof. Norman F. Schneidewind, Code 54Ss Naval Postgraduate School Monterey, California 93940	1
4.	Superintendent ATTN: Prof. M. G. Sovereign, Code 74 Naval Postgraduate School Monterey, California 93940	2
5.	Superintendent C3 Curricular Officer, Code 39 Naval Postgraduate School Monterey, California 93940	1
6.	Director, National Security Agency ATTN: Mr. George M. Hicken CCINS Project Management Office 9800 Savage Road Ft. George G. Meade, Md. 20755	2
7.	Chemical Abstract Service ATTN: Dr. Ronald L. Wigington Director, Research and Development P.O. Box 3012 Columbus, Ohio 43210	1
8.	Massachusetts Institute of Technology ATTN: Mr. Richard Marcus Laboratory for Information and Decision Systems Cambridge, Mass. 02139	1
9.	Environmental Protection Agency ATTN: Dr. Sidney Siegel, TS-777 401 M St., S.W. Washington, D.C. 20460	1
10.	USS Enterprise (CVN-1) ATTN: Lt. B. French Operations Department FPC San Francisco 96635	1
11.	Commandant of the Marine Corps Code CCA ATTN: Capt. R. Graham Headquarters, United States Marine Corps Washington, D.C. 20380	1

12. Majcr Tae Nam Ahn 1
Computer Center
P.C. BCX 77
GongNeung_Dong, Dobong_Gu
Secul 130-09, Korea

13. Hellenic Navy 1
ATTN: Cdr. Cosmas Charpantidis
General Naval Staff
Hclargcs
Athens, Greece

14. Directcr, National Security Agency 5
ATTN: J.B. Kim, S66
9800 Savage Road
Ft. George G. Meade, Md. 20755

202204
Thesis
K41725 Kim
c.1
An investigation into the coupling of interactive and batch network services in COINS.